

DECISION SUPPORT SYSTEM FOR SEED
SELECTION USING SPATIALLY-
REFERENCED SOIL DATA

By

PHILLIP A. MARTIN

Master of Science in Computer Science
Oklahoma State University
Stillwater, OK
2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2009

DECISION SUPPORT SYSTEM FOR SEED
SELECTION USING SPATIALLY-
REFERENCED SOIL DATA

Thesis Approved:

Dr. John Chandler

Thesis Adviser

Dr. Douglas Heisterkamp

Dr. Nophill Park

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express sincere thanks to my advisor, Dr. John Chandler for all his help, supervision, and advice throughout the thesis process and in the many classes in which he shared his knowledge with me. I also wish to extend thanks to my other committee members Dr. Douglas Heisterkamp and Dr. Nohpill Park, who have inspired me and helped me to achieve this milestone, and have similarly provided me with excellent classroom experiences. I would further like to extend thanks to Dr. Blayne Mayfield who also taught me many things, and was kind enough to lend his aid when I needed it.

I would further like to thank my wife, Jennifer Martin for pushing me to achieve my goals when things were difficult, and my parents Steve and Karen Martin, for their support, encouragement, and understanding throughout the process.

Finally, I would like to thank the owners and staff of SST Software, who were willing to support my desire to receive a graduate degree, and gave me the opportunity to make it happen.

TABLE OF CONTENTS

Chapter	Page
CHAPTER I - INTRODUCTION	1
The Problem.....	1
Sources of Data.....	1
Proposed Solution.....	1
Previous Work	1
Definitions of Terms.....	2
CHAPTER II - METHODOLOGY AND TECHNOLOGY REVIEW.....	3
Solution Methodology	3
Part 1: Relational Databases	3
Part 2: Soil Data and the GIS System.....	4
Part 3: The Expert System Shell	5
Part 4: End User interaction.....	6
Part 5 – The system as a whole.....	7
CHAPTER III - DATA LAYOUT	9
Seed Data	9
Soils Data.....	12
CHAPTER IV - COMPONENT INTERACTION.....	15
Windows App with end user.....	15
Windows App with SQL Server	21
Windows App to OGC Services	22
Windows App to CLIPS	23
Within CLIPS.....	24
CHAPTER V - RESULTS.....	28
Run 1 - No Miscellaneous options.....	28
Run 2 - Some options set	30
CHAPTER VI - SUMMARY	32

Conclusions.....	32
Future/Suggested Work	32
REFERENCES	34
APPENDIX A - CLIPS RULES FILE.....	35
APPENDIX B - CUIManager Class	43

LIST OF TABLES

Table 1 - Fact Modification Values	27
Table 2 - Selected Soils.....	29
Table 3 - Seed recommendations, no miscellaneous facts.....	29
Table 4 - Top three seed details	30
Table 5 - Ratings with some liberty, rootworm	30
Table 6 - Seed Details for top four varieties with options set.....	31

LIST OF FIGURES

Figure 1 - Application Component Interactions	7
Figure 2 - Seed Entity-Relationship Diagram.....	10
Figure 3 - Relevant Soil Data.....	13
Figure 4 - Area selection screen.....	16
Figure 5 - Select from preset areas	17
Figure 6 - Intersecting soils preview.....	17
Figure 7 - Field options.....	18
Figure 8 - Seed Options	19
Figure 9 - Results form initial interface	20
Figure 10 - Results form, final display	21

CHAPTER I

INTRODUCTION

The Problem

Seed genetics presently play a very important role in the field of precision agriculture. By using seeds that have been tailor made to work well in specific conditions, yields can be improved on land previously considered problematic [Wang]. The drawback to this method comes into play when the wrong type of seed is used for a given set of field conditions, which can actually lead to reduced yield and/or increased chance of disease in the crop. Soil type is one of the predominant characteristics of any field, as the type of soil directly affects many variables important to seed selection, such as the amount of water available, the rate of drainage, or the amount of erosion. The process of selecting a seed for a specific group of soils comprising a growing area would require special knowledge of the soils being planted, the potential hazards these soils present, and what types of seeds can combat these hazards. While this process can produce good results, it is laborious and requires a fair amount of communication between the grower, the soil specialist, and the seed specialist.

Sources of Data

The potential exists to use publicly available data to help solve this problem. Seed companies provide information about the relative effectiveness of their many

varieties and hybrids in dealing with adverse growing conditions and a multitude of diseases. An example is the Pioneer seed company, which publishes ratings for common diseases and crop attributes in their catalogs and online variety searches. Further, the United States government provides soil information for all arable counties in the continental states, in a standardized format known as SSURGO. There exist other sources of data that can be helpful, such as satellite imagery to measure crop health or weather information from NOAA (National Oceanic and Atmospheric Association), which are both helpful in crop planning, but will not be addressed herein. By applying expert knowledge with the available information, a system can be created to aid in recommendation of suitable seeds.

Proposed Solution

I propose that a Decision Support System (DSS) for recommendation of seeds be created and maintained, leading to simpler decision making and improved crop yield. DSS are "interactive computer-based systems that help decision makers utilize data and models to solve unstructured problems" [Sprague]. The system will involve a Windows forms application for end user activities, a set of spatially-referenced soil data used with a GIS system, a database for storage of seed data, and an expert system shell (CLIPS) to apply knowledge to the collections of data.

Previous Work

I am unaware of any previous Decision Support System designed to recommend seed varieties, although there has definitely been research regarding both decision support

systems and proper seed placement. The intersection of these disparate technologies and disciplines appears to be unique.

Definitions of Terms

- OGC – The Open Geospatial Consortium – a nonprofit consortium for defining standards for geospatial and location-based services.
- WFS – Web Feature Server – a standard created by the OGC for an XML-based web service that can transmit geospatial information in one of several standardized XML formats.
- WMS – Web Map Server – a standard created by the OGC for an XML-based web service that allows a user to retrieve stylized renderings of subsets of geographic vector and raster data in a multitude of image formats.
- NRCS – National Resources Conservation Service – A federal agency tasked with helping private land owners conserve soil, water, and other natural resources.
- SSURGO - Soil Survey Geographic Database – A standard data representation for soils data provided by the NRCS.
- CLIPS - C Language Integrated Production System – An expert system shell originally developed by NASA at the Johnson Space Center.
- DSS – Decision Support System - A software system that aids users in making decisions about unstructured problems by applying expert knowledge and rules to sets of facts.
- RDBMS – Relational Database Management System – A system used to store, maintain, and query data utilizing concepts of Entities and Relationships in order to organize data in a logical and efficient manner.

CHAPTER II

METHODOLOGY AND TECHNOLOGY REVIEW

Solution Methodology

The solution to this problem involves four distinct parts. Firstly, a relational database will be created to house information about specific seed varieties. The second part will be a set of spatially-referenced soils data, loaded into a relational database and accessed via an OGC GIS system consisting of both a Web Feature Server (WFS) and a Web Map Server (WMS). The third part is an expert system shell that interprets sets of facts and rules, used to determine suitable seeds based on the applicable soil types and settings specified by the end user. The final part is a custom application, designed for the end users of the system. This application will allow interactivity with the Decision Support System, providing a final set of recommendations. By integrating all these disparate portions of the system into a cohesive whole, an end user will be able to interact indirectly with large amounts of data and a set of domain-specific knowledge that may not otherwise be so readily available.

Part 1: Relational Databases

The proposed solution will make use of two separate databases to house the needed data. First, a database must be created that contains all the needed information about each seed variety. Second, another database will be created to contain soil data.

This database will exactly mirror the structure set forth by the NRCS for their SSURGO datasets. This should preserve pre-defined relationships and allow those familiar with the standard to easily use and update the data. For the implementation of this system, SQL Server Express has been selected as the target relational database management system (RDBMS). SQL Server is a product maintained and sold by the Microsoft Corporation, originally part of a joint venture with the Sybase Corporation. SQL Server is a popular RDBMS, with a large installed base and advanced features that make it attractive to enterprise customers using Microsoft-based platforms. As such, it is widely available in many environments. SQL Server supports many data types natively, including spatial and XML data, making it an excellent choice for the problem at hand. Our selected software, SQL Server Express, is a free-to-use version of the full product, making it suitable for independent developers and small businesses, which will include many of the target users of this system.

Part 2: Soil Data and the GIS System

The USDA provides soil data for all agriculture producing counties in the United States via the Natural Resources Conservation Service (NRCS). This data is regularly updated and maintained by the NRCS in the form of the Soil Survey Geographic Database (SSURGO). The data itself are distributed as a collection of text files containing comma-separated values, one set per soil survey area – typically a county. The NRCS provides an optional Microsoft Access database template that will load all of the individual text files into a set of relational tables. The spatial data is typically delivered in the industry-standard ESRI Shapefile format. These data sources may then be loaded into an OGC-compliant Web Feature Server and Web Map Server to handle

GIS operations such as geographical intersections and spatial queries. SSURGO data will also be loaded into the aforementioned SQL Server system for more complex soil profile queries used to assert facts about the selected soil types in the expert system. The GIS platform used for the system is Geoserver. Geoserver is an open source implementation of several OGC-specified standards, including the Web Feature Server (WFS), the Web Map Server (WMS) and the Web Coverage Server (WCS) standards. Geoserver is the OGC reference implementation for both WFS and WCS, as well as a certified-compliant WMS implementation. Written in Java, Geoserver seeks to encourage interoperability between systems, using standard formats for geographic data. Geoserver leverages several open source libraries to fulfill all of its goals, and the heart of the GIS engine is built on Geotools, an open source Java GIS toolkit.

Part 3: The Expert System Shell

The project makes use of an embedded version of the CLIPS expert system shell. CLIPS (C Language Integrated Production System) was a product of NASA's Johnson Space Center in 1985. CLIPS was created as a way to deliver an expert system not based on LISP, as LISP provided several barriers to entry due to high cost, lack of support on common hardware, and poor integration with other languages. After finding no satisfactory offerings from commercial vendors, NASA decided to develop its own system. Thanks to wide availability and low cost, CLIPS has become a widely used tool for expert systems. CLIPS can be defined more precisely as a forward-chaining rule language based on the Rete algorithm. Later enhancements to CLIPS added support for procedural programming and object-oriented paradigms. Since CLIPS is openly available, many derivative versions have been created for use with many languages and

architectures. For this implementation, a version of CLIPS modified to integrate with the Microsoft .NET Framework created by MommoSoft (<http://www.mommosoft.com/>) was selected. This expert system will consist of sets of rules and facts to integrate the knowledge of experts in the fields of soil science and seed genetics to determine suitable selections of specific seed varieties for the applicable soil types. In our implementation the facts will be dynamically generated based on the soil polygons found to intersect with the geographic area specified by the user, and the users' criteria for seed selection. In this way, the expert system will have a pre-culled list of facts to work with, in order to select proper matches for the user more efficiently.

Part 4: End User interaction

Finally, a method for the end user to interact with the DSS must be created. The end users need the ability to select what soil types they need information for, the ability to set options about certain seed attributes that are pertinent to the selection process but not directly related to the soils being planted, and finally a method to find out what applicable seeds match these criteria. A graphical user interface (GUI) is desirable for easier interaction with non-technical staff, and the ability to interact easily with the previously mentioned technologies is paramount. The system presented was created with the C# programming language, using Microsoft's Visual Studio 2005 development environment and compiler. C# is a general purpose, modern, object-oriented language, with syntactic similarities to Java and C++. C# has been standardized by the ECMA and ISO organizations (ECMA-334 and ISO/IEC 23270 respectively). C# is partially compiled and partially interpreted, using the Microsoft .NET Runtime for host system execution. Using C#, a GUI-based desktop application was created that will query the

aforementioned SQL Server databases, manage the expert system shell, track user settings, and support user interaction with all necessary parts of the system as a whole.

Part 5 – The system as a whole

A notable feature to the organization of this system is the end user application. As illustrated in Figure 1, this portion of the system is central to all other operations. The end user application is responsible for sending data to all subsections of the system, and maintaining the results between calls to other subsections. As well as being the users' point of contact with the DSS as a whole, the application actually manages all the distinct parts of the system.

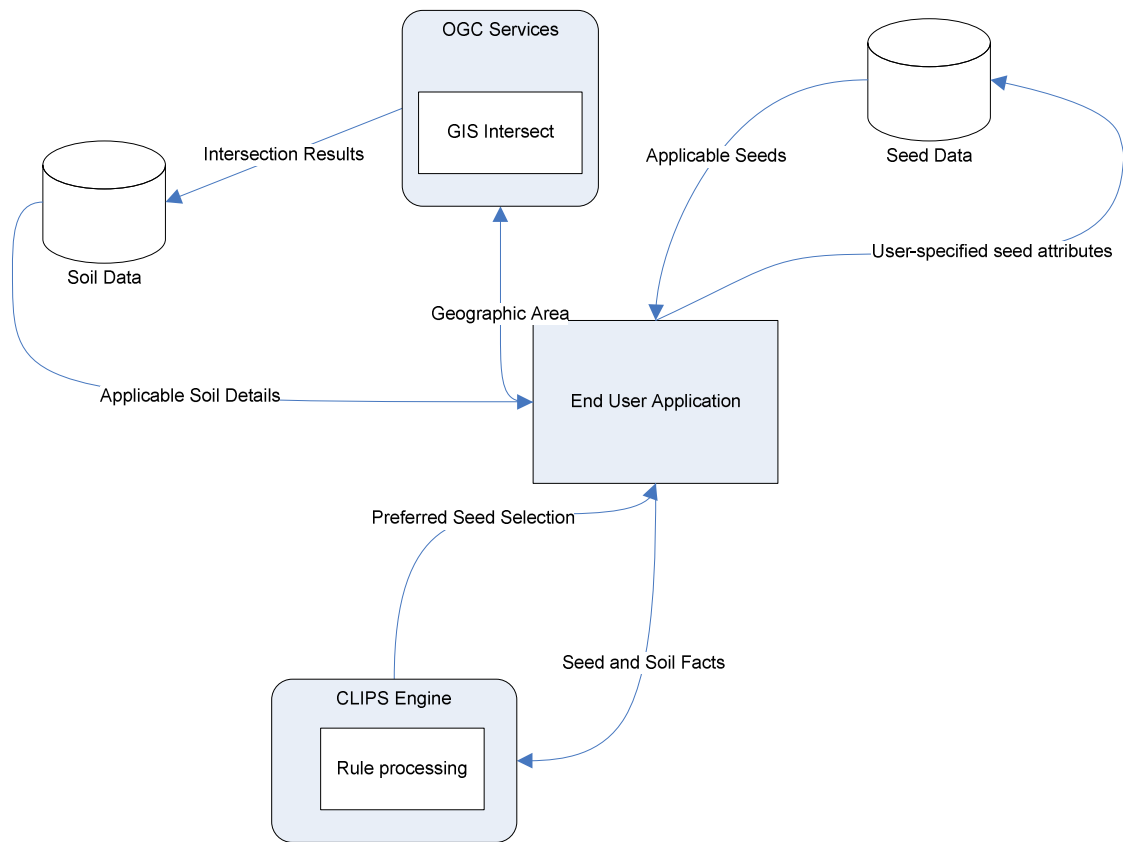


Figure 1 - Application Component Interactions

While parts of the system, such as the CLIPS Engine and the OGC services have very distinct roles to play, and may actually perform some of the more complex operations of the system within their own boundaries, the combination of the diverse subsystems is what allows for the flexibility to automate all the aspects of the mundane tasks an expert would have to perform in order to get the information required to even begin evaluating a problem. Since neither an expert nor the end user needs to examine charts, maps, or tables in order to perform the basic tasks, the system is suitable for use by most potential end users.

CHAPTER III

DATA LAYOUT

Seed Data

The data describing the seeds are fairly simplistic in nature. While there can be a large amount of information, and some relational structure is helpful for ancillary attributes, the bulk of the interesting seed data deals with resistances to diseases and several important crop attributes, with one rating per variety. These values are integers from one to nine, so a single table with many columns is used to hold all characteristic information. In planning for multiple crop types in the future of the application, the type of crop has been linked via a foreign key to a parent table in the current schema for seeds. This was done to show that the same table can be used for different types of crops, should we wish to expand this system to support other crops such as soybeans, wheat, etc. However, to illustrate the operation of the system, we will consider only corn in the rest of this section.

The seeds database consists of two tables, Crop and Seeds. The two tables are related via the Foreign Key FK_Seeds_Crop. Each seed may be only one crop, so this foreign key defines the relationship between the column “CropID”, which is the primary key in the table “Crop” with a single reference to the values defined in that table to the rows in the table “Seeds”. Definitions for the columns follow:

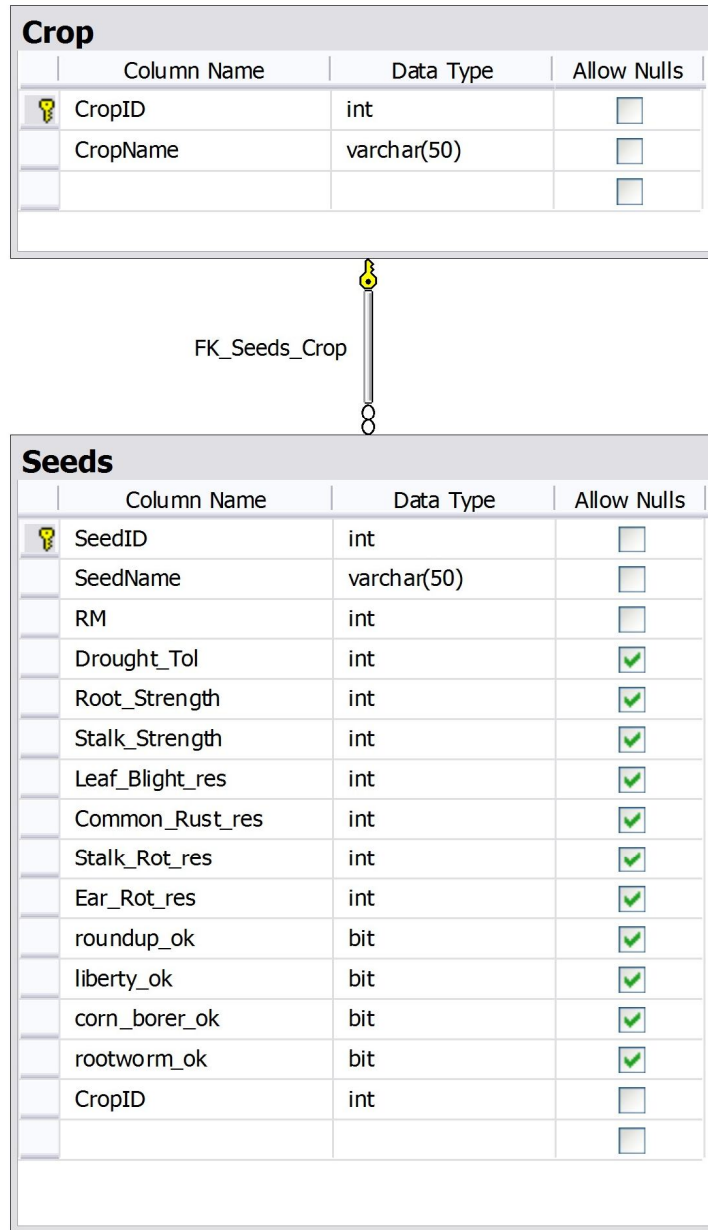


Figure 2 - Seed Entity-Relationship Diagram

Table Crop:

CropID – The primary key for this table, an automatically increasing integer sequence value.

CropName – The name of the crop, such as “Corn” or “Soybeans”, a variable length character field with a maximum of 50 characters.

Table Seeds:

SeedID – The primary key for this table, an automatically increasing integer sequence value.

SeedName – The name of the seed variety, a variable length character field with a maximum of 50 characters.

RM – The relative maturity of the variety, an integer value that corresponds approximately to the number of days of good growing conditions required for this seed to reach maturity.

The following characteristics are represented as integers with possible values of 1 through 9. Lower values mean poorer performance, and each rating is assigned by the seed company as an estimate of that varieties’ performance.

Drought_Tol – Drought tolerance, how well the seed can handle dry conditions.

Root_Strength – The strength of the root system created by the seed.

Stalk_Strength – The strength of the corn stalk.

Leaf_Blight_res – Resistance to leaf blight, a common disease.

Common_Rust_res – Resistance to common rust, another common disease.

Stalk_Rot_res – Resistance to stalk rot, another common disease.

Ear_Rot_res – Resistance to ear rot, another common disease.

The following columns represent genetic traits engineered into a variety to help it deal with possible field conditions. These traits either exist or they do not, so they are stored as a single bit representing a Boolean value of false (0) or true (1):

Roundup_OK – This seed has genetic traits to prevent it from being susceptible to

Roundup, a common herbicide with the active ingredient glyphosate.

Liberty_OK – This seed has genetic traits to prevent it from being susceptible to Liberty,

another common herbicide with the active ingredient glufosinate-ammonium.

Corn_borer_OK – This seed has a genetic resistance to the European corn borer, the

larvae of a moth that tunnels through the ears and stalks of corn plants.

Rootworm_OK – This seed has a genetic resistance to rootworms, a beetle whose larvae

feed on the roots of corn and adults that feed on the ears.

CropID – a Foreign Key reference to the crop table.

Soils Data

The soils database consists of the schema defined by the USGS and the data provided for the selected counties. While the system contains the entire schema, only a small portion of the relevant data is used by the DSS. In Figure 3, only the relevant columns from the relevant tables are displayed.

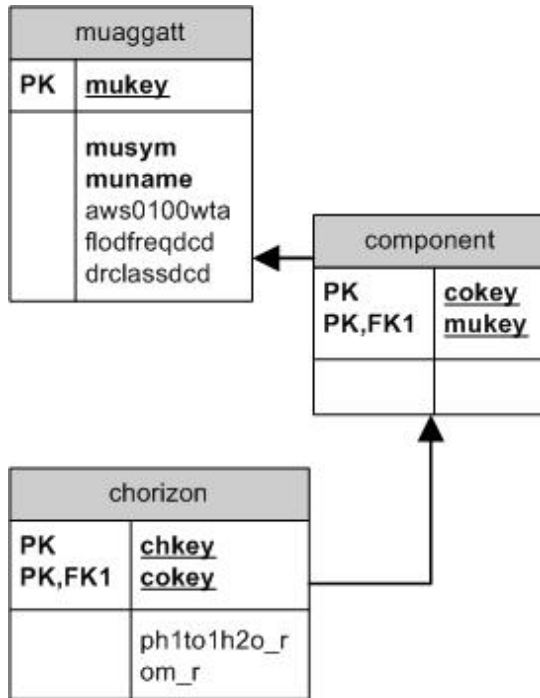


Figure 3 - Relevant Soil Data

From the soil data, we are making use of three tables, two of which contain necessary data. Following is a description of the tables and the columns within.

Table muaggatt:

The muaggatt table contains aggregate values for each unique mukey. These aggregate values summarize the entire soil profile for all the individual components that make up an individual soil or soil complex.

Mukey – the primary key of the table, and the value we’ll receive from the geoserver WFS service.

Musym – The symbolic name of the soil used on a map. Typically, this is an integer number or a short string of letters for special geographic features, such as water and landfill.

Muname – The full name of the soil or soil complex.

Aws0100wta – the amount of available water in the top 100 centimeters of soil.

Flodfreqdcd – a string that tells us the frequency of flooding on this soil.

Drclassdcd – a string telling us the drainage class of this soil.

Table component

The component table lists, for an aggregate soil complex, the different components that constitute that soil. A soil has one or more components. In this system, we won't examine the data at the component level, but will simply use it to find the primary component of an aggregate soil complex, and then determine some averages of that primary component's horizons. A horizon is a layer of soil that has physical characteristics that differ from the layers above and below [Soil Survey Division Staff].

Table horizon

The horizon table lists different horizons for the individual soil components. At different levels of depth, a soil component may exhibit different properties. By examining the average values of some of these columns, we will find some useful information about the primary soil components.

Chkey – the primary key of this table.

Cokey - the foreign key to the components table.

Ph1to2h20_r – the representative value for the pH reading of this soil horizon.

Om_r – the representative amount of organic matter in this soil horizon.

CHAPTER IV

COMPONENT INTERACTION

Windows App with end user

The primary method of interaction between the decision support system and the end user is through the use of a Windows forms application written in C#. The goal of the window application is to provide a simple method of interaction with all the distinct pieces of the system. The application is arranged by task, so that a single form allows the user to interact with a single aspect of the system as a whole, be it a component of our system or a logical division the user would have in their own decision making process. While the users are allowed to work on the individual forms in any order they wish, a preferred order is provided and easily navigated through using “Back” and “Next” buttons, in an attempt to use the familiar concept of a Wizard. Each step is reviewed here in detail.

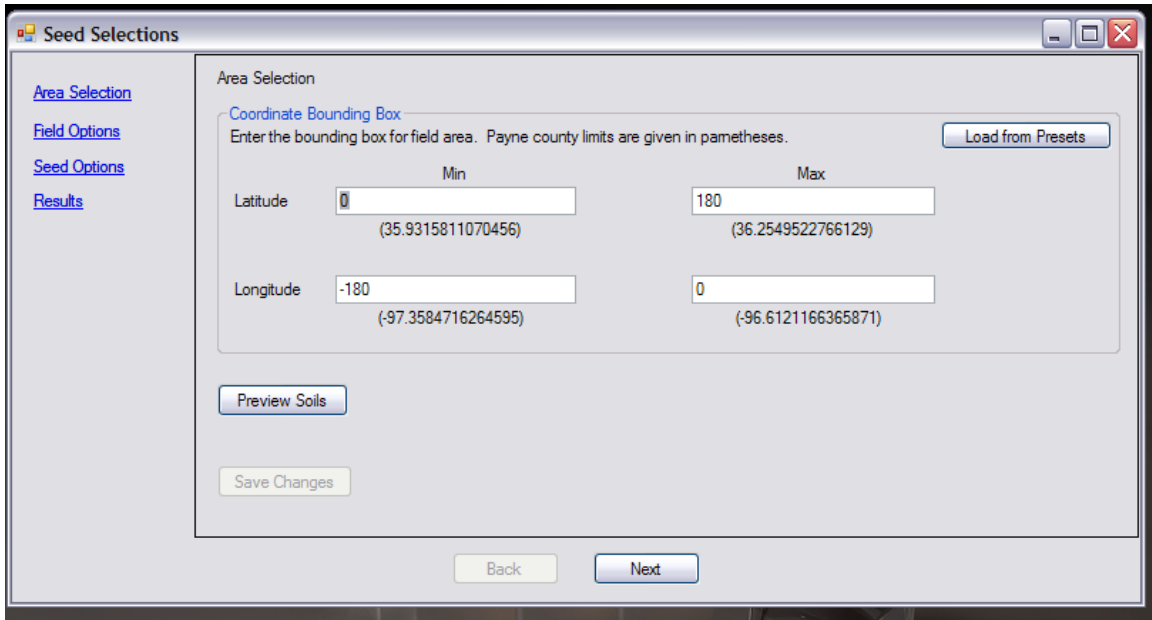


Figure 4 - Area selection screen

The first screen the user is presented with (shown in Figure 4) allows the user to specify the geographic bounding box to investigate for soil information. All coordinates given here are specified in decimal degrees of latitude and longitude, with examples provided to illustrate this point. The user may also select from the presets (Figure 5) provided. The user may also choose to preview the soils that intersect with the specified bounding box (Figure 6). Once the user is satisfied with the results, he or she saves the changes and selects the “Next” button to continue to the next step.

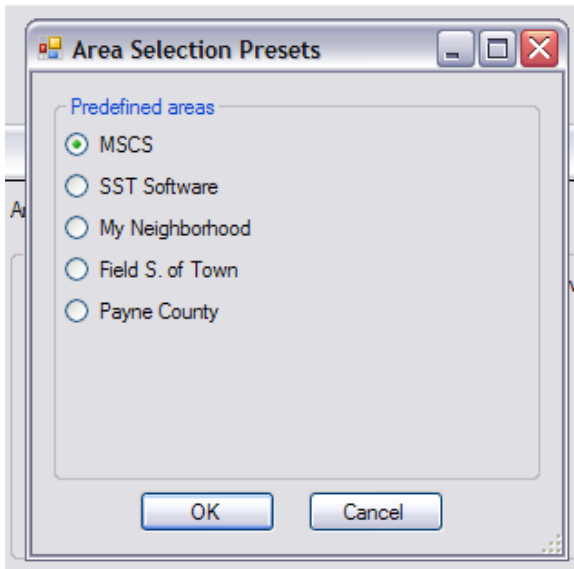


Figure 5 - Select from preset areas

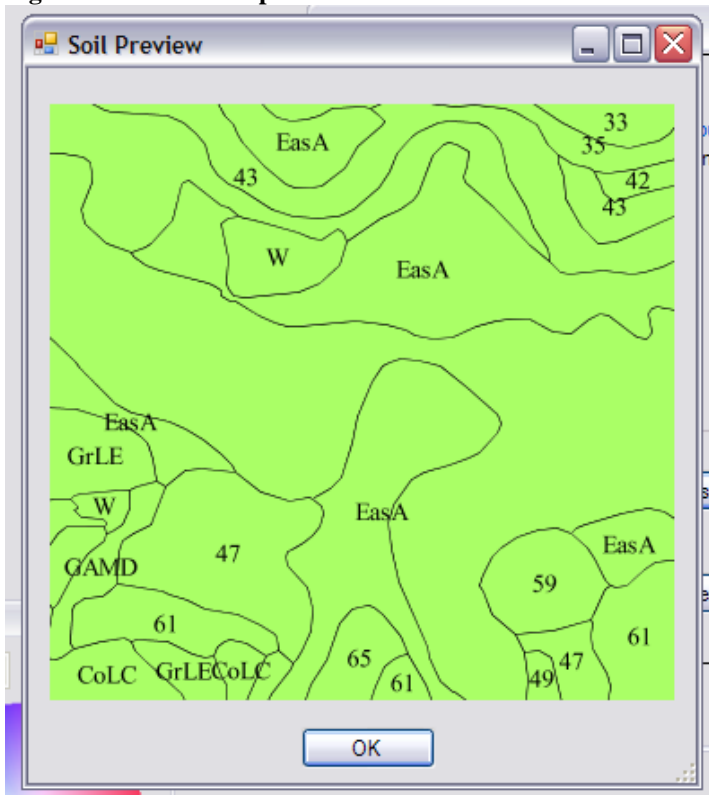
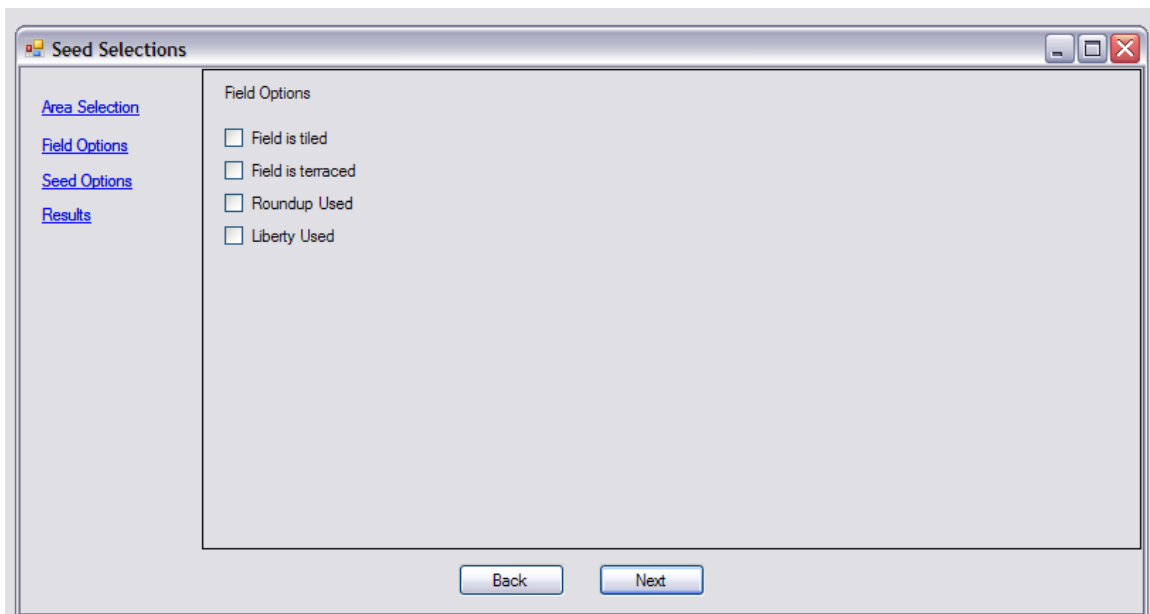


Figure 6 - Intersecting soils preview

Following the saving of a selection of a geographic area for placement, the user will enter options about the user's field. The goal of the field options screen (Figure 7) is to gather data for the expert system about the geographic area that cannot be derived

solely from the soil data as well as the gathering of information on treatments applied to the field. The current system allows the user to set switches related to both physical manipulation of the soil and the application of pesticides to the field. The four presented options are field tiling, field terracing, use of Roundup and use of Liberty. Tiling is the practice of installing pipes below the surface to allow water to drain from the soil - this is helpful in very wet soils. Terracing is a method of soil conservation to help reduce erosion. Roundup and Liberty are herbicides used to control weeds and other invasive plants. Changes to the checkboxes on this form are stored immediately, without the need to interact with a “Save” or “Commit” button.

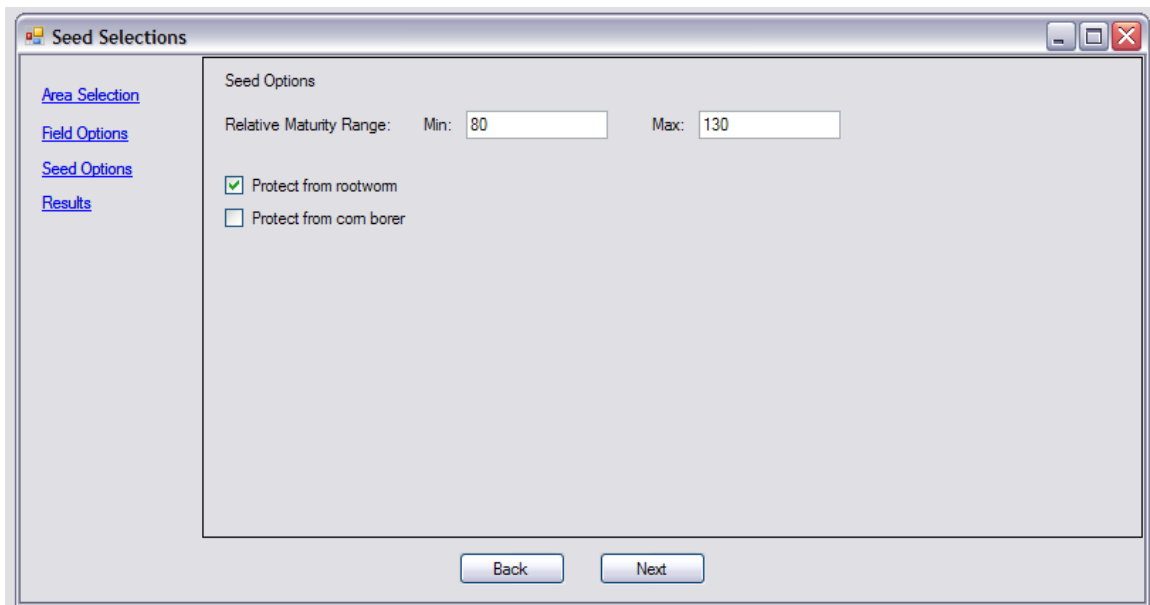


The screenshot shows a window titled "Seed Selections" with a standard Windows-style title bar. On the left side, there is a vertical menu with four blue hyperlinks: "Area Selection", "Field Options", "Seed Options", and "Results". The "Field Options" link is currently selected. The main content area of the window is titled "Field Options" and contains four unchecked checkboxes, each followed by a label: "Field is tiled", "Field is terraced", "Roundup Used", and "Liberty Used". At the bottom center of the window, there are two buttons: "Back" and "Next".

Figure 7 - Field options

Following the field options UI, the user is presented with the seed options form (Figure 8). This form allows the user to enter data that will both help determine the list of seed facts to send to the expert system and pass the expert system some facts to aid in the seed selection process. Included in the form are boxes for both a minimum and

maximum value for relative maturity, as well as a request for corn borer and/or rootworm protection. Relative maturity is related to the number of growing days needed for the crop to reach maturity. Generally, the greater the relative maturity, the greater the weight or yield for a mature crop. However if the grower selects a crop with a relative maturity that is too large, the crop may not reach full maturity before the growing season ends. Rootworms and Corn borers are common pests that can ruin corn crops, so if a grower believes they may have problems with these pests, then a hybrid with an inbred resistance to the pest may be preferable to help minimize damage. The values given in this form are saved when changed, so the user has no need to interact with a “Save” or “Commit” button before leaving the form.



The screenshot shows a window titled "Seed Selections" with a sidebar on the left containing links for "Area Selection", "Field Options", "Seed Options", and "Results". The "Seed Options" section is active and contains the following fields and controls:

- Relative Maturity Range: Min: Max:
- Protect from rootworm
- Protect from corn borer

At the bottom of the window are two buttons: "Back" and "Next".

Figure 8 - Seed Options

The final form for the user to interact with is the results form (Figure 9). Upon first load the user will be presented with a button to begin the recommendation process.

When clicked, progress notifications are sent to the screen while the application begins to interact with its internal classes, the database, and the expert system shell.

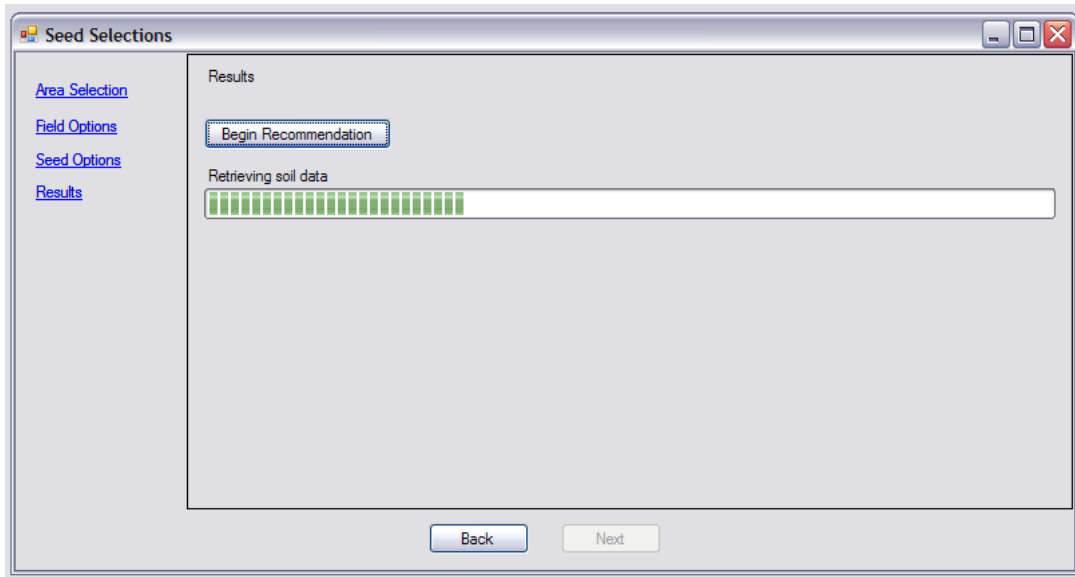


Figure 9 - Results form initial interface

After the expert system shell has performed its duty and assigned each seed hybrid a relative rating value, the results of these ratings are displayed to the user, in descending order (Figure 10). In order to show what soils were used in the classification process, a table displaying the symbols and long-form names of the soil types is also shown to the end user.

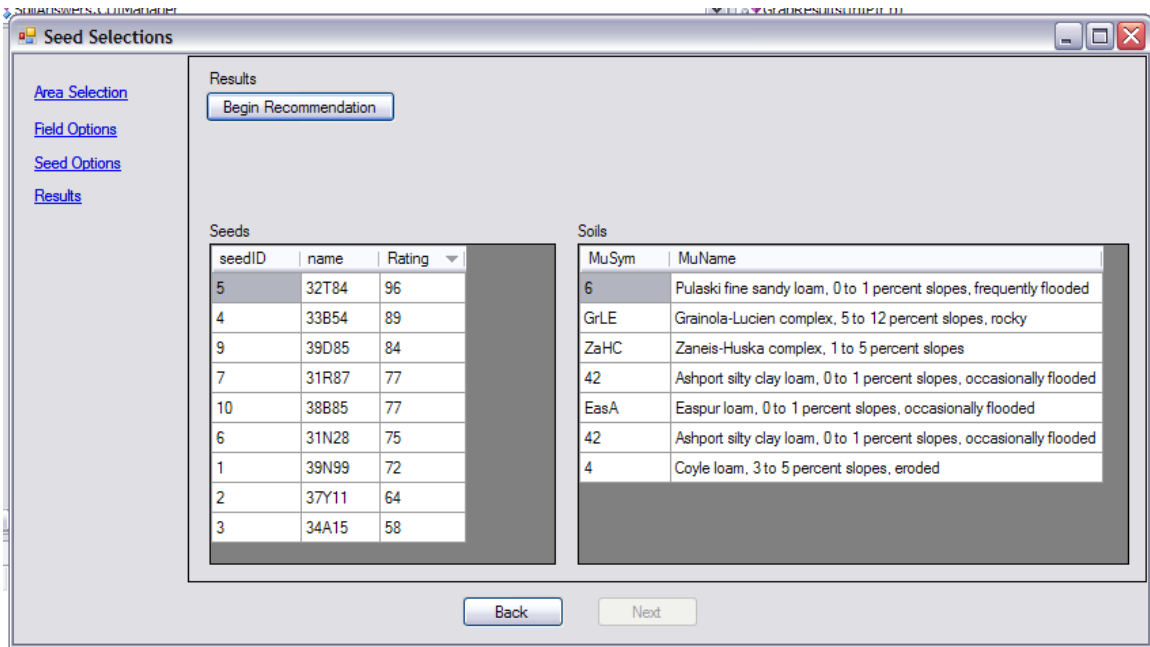


Figure 10 - Results form, final display

Windows App with SQL Server

Since information about both seeds and soils are stored in the SQL Server relational database, the application must interact with those databases in order to provide necessary data to the expert system. All interaction with SQL Server happens in the results form.

When the user decides to begin the recommendation process the application will first query SQL Server for information about the soils that were selected in the area selection form. For each mukey selected in the area selection form, a new CSoilFact class is instantiated. When the user clicks the Begin Recommendation button on the results form, each of those instantiated objects is examined for its MuKey, which is then used as an argument in the where clause of an SQL query to discover more attributes of the soil or soil complex.

After all the information about the selected soils has been read from the soils database, the application will next query the seeds database to find candidate seeds for the expert system to rate. In order to determine the list of seeds, the user's selection for minimum and maximum relative maturity are used in the "where" clause of an SQL query. Since the user has selected the relative maturity range, the decision was made to return only the seeds within that range, rather than have the expert system eliminate them. Sending less data to the expert system helps to reduce the amount of processing needed to reach a conclusion. Further, it is a decision that should be based on the grower's logistical needs and the field's climate, not simply from soil data. For each seed row that falls within this given range, a new CSeedFact class is instantiated, and its data is filled from the database directly after creation time.

Windows App to OGC Services

In order to determine the types of soil that lie within the geographic boundaries specified by the user, two separate Open Geospatial Consortium (OGC) Services are used with a single dataset. First, a Web Feature Server (WFS) is used to retrieve the list of soil features that intersect the bounding box of minimum and maximum latitudes and longitudes. The WFS returns an XML document that the application parses to find all the MuKeys returned. To support the preview screen, a Web Map Server (WMS) is used to create an image of the specified bounding box, displaying the intersecting portions of the soil polygons, with the MuSym column from the data used as a label for each polygon. The interaction between the application and both OGC servers happens on the Area Selection form. The interaction between the two is handled via HTTP Requests to the OGC servers, and an HTTP Response of either plain text XML or binary image data

returned from the WFS and the WMS respectively. The source data used by both services consists of an ESRI Shapefile (.shp) with a matching spatial index file (.shx) and a small subset of the SSURGO data in a flat binary data file (.dbf). The MuKeys, MuSyms, and MuNames listed in the dbf and referenced by the shp correspond exactly to the SSURGO dataset stored in SQL Server.

Windows App to CLIPS

The Windows application reaches the end goal of seed recommendations by combining all the data received from the various other parts of the decision support system and creating facts to send to the expert system shell. CLIPS was selected as the expert system shell for this application, due to its widespread availability and well-documented programming interface. Throughout the course of running the application, various user-entered and programmatically retrieved pieces of data are stored and maintained until the user requests the results from the expert system shell.

The application has a class that manages all the user-selected data values, and maintains the list of facts, as well as the reference to the CLIPS engine object. This class – CUIManager (Appendix B) – will initialize the CLIPS engine, load it with the ruleset file, handle the assertion and retraction of facts, and handle events fired by the shell.

Upon activation of the expert system shell, these various sources of data are used to derive one of three types of facts. The application has provided three templates for facts in the form of classes derived from the Clips.NET Fact class: CSoilFact, CSeedFact, and CMiscFact. CSoilFact and CSeedFact benefit from a Clips-defined deftemplate, on top of the C# class definition. The deftemplate construct allows the developer to define a fact with a specific collection of slots (variables) to simplify

development with those facts. CMiscFact was created as a catch-all plain-text fact, so that miscellaneous facts that need to be asserted, but that don't benefit from the slotting mechanism, can be rendered easily by the application.

The expert system shell will assert all the facts defined by the application and, using the rules defined in rules.clp (Appendix A), will perform the analysis needed to apply a ranking to each seed hybrid. The Clips.NET library used also provides functionality to trigger a .NET event, based on the ClipsFunction delegate, to raise an event when fired from the right-hand side (RHS) of a CLIPS rule. Using this functionality along with the CLIPS concept of salience, we can specify a rule to update the UI when all CLIPS seed processing has been completed.

Within CLIPS

Once CLIPS has been loaded with the rules for determining ratings for soil and seeds, along with the applicable facts for the soils, seeds, and options of interest to the user, the shell is given the run command, and the inference engine begins executing.

Both the seed facts and the soil facts have a rating related to each fact. For the seed facts, the rating slot is named "suitRate" which is of integer type with a default value of 50. For the soil facts, the rating slot is named waterRating, which is also of integer type and has a default rating of 50. These ratings values are increased or decreased based on other values found within the set of facts, or by the user's request for special considerations of pests and field conditions.

The execution order of the rules is roughly controlled via the concept of "salience" within the system. Salience applies a preference for execution to the individual rules, so that rules with a higher salience value execute before those with a

lower salience value. In the rule set used in this implementation we break our rules down into five levels of salience, with an extra sixth level that does nothing but notify the User Interface that the inference engine is finished working. The top level includes rules that modify the ratings of the seeds and soils by user requests for protection from pests, resistance to pesticides, and field options. The second level modifies the waterRating slot of the soil facts based on available water, flood frequency, and drainage class. After the soils have been classified, the next level of salience executes to determine the soil with the worst water rating. This minimum value ensures that the worst soil is accounted for in our classifications. The next level of salience will modify the suitRate slot of the seed facts based on the seed attributes and the worst waterRating discovered in the previous level of salience. The major categories investigated in this step are drought tolerance and root strength. The final level of salience modifies the suitRate slot of seed facts based on the remaining minor seed categories.

The seed's rating is influenced by what we consider to be major categories – which increase the rating by larger amounts, and minor categories, which increase the rating by smaller amounts. The minor categories are used to help break ties between seeds that have equal ratings in the major categories but exhibit different characteristics elsewhere. Instead of a set amount of movement for each possible value in a minor category, the rating value for that category is added to the seed fact's rating. So if a seed rates the same based on the worst water rating, but has a higher resistance to a disease, such as Common Rust, it can be recommended over a similar seed with poorer disease resistances.

A summary of the method of modifications for both soil and seed facts are performed is given in Table 1. Note that the test values for seed characteristics range from 4 to 8. While investigating the seed ratings given by the Pioneer seed company over a range of different geographic areas, the ratings were predominately within these values, with very few ratings of 3 or 9 given. No ratings of 1 or 2 were found. Because of this, the tests are run against the truly used values of 4 through 8.

Fact Type	Attribute Checked	Test Value	Modification
Seed	cornBorer	True + User Request	suitRate + 10
	Rootworm	True + User Request	suitRate + 10
	Roundup	True + User Request	suitRate + 10
	Liberty	True + User Request	suitRate + 10
	droughtTol	$0 < \text{droughtTol} \leq 4$	$\text{suitRate} + (\text{wwr} - 70)$
	droughtTol	$\text{droughtTol} = 5$	$\text{suitRate} + (\text{wwr} - 65)$
	droughtTol	$\text{droughtTol} = 6$	$\text{suitRate} + (\text{wwr} - 60)$
	droughtTol	$\text{droughtTol} = 7$	$\text{suitRate} + (\text{wwr} - 55)$
	droughtTol	$\text{droughtTol} \geq 8$	$\text{suitRate} + (\text{wwr} - 50)$
	rootStr	$0 < \text{rootStr} \leq 4$	$\text{suitRate} - (\text{wwr} - 40)$
	rootStr	$\text{rootStr} = 5$	$\text{suitRate} - (\text{wwr} - 45)$
	rootStr	$\text{rootStr} = 6$	$\text{suitRate} - (\text{wwr} - 50)$
	rootStr	$\text{rootStr} = 7$	$\text{suitRate} - (\text{wwr} - 55)$
	rootStr	$\text{rootStr} \geq 8$	$\text{suitRate} - (\text{wwr} - 60)$
	Other characteristics	$0 \leq \text{characteristic} \leq 9$	suitRate + slot value

Soil	User Entered Tiled	User Entered	waterRating - 5
	User Entered Terraced	User Entered	waterRating + 5
	availWater	0 < availWater < 13	waterRating - 20
	Availwater	Availwater > 16	waterRating + 20
	floodFreq	floodFreq = 0	waterRating - 10
	floodFreq	floodFreq = 2	waterRating + 5
	floodFreq	floodFreq = 3	waterRating + 10
	drainageClass	0 <= drainageClass =< 1	waterRating - 10
	drainageClass	drainageClass = 2	waterRaing - 5
	drainageClass	drainageClass = 3	waterRating - 3
	drainageClass	drainageClass = 5	waterRating + 5
	drainageClass	drainageClass = 6	waterRating + 10

Table 1 - Fact Modification Values

CHAPTER V

RESULTS

Run 1 - No Miscellaneous options

An example run was performed without any field or seed options selected, in order to focus solely on the soil classification mechanics of the expert system:

Minimum latitude: 36.1306

Maximum latitude: 36.13274

Minimum longitude: -97.12806

Maximum longitude: -97.12261

Field is tiled: false

Field is terraced: false

Roundup used: false

Liberty used: false

Relative maturity range: 95-120

Protect from rootworm: false

Protect from corn borer: false

The soils included in this bounding box are shown in Table 2. Note that the Ashport silty clay loam soil complex is listed twice. This particular geographic area has two separate instances of this soil with different MuKey values, meaning that some characteristics of the soil are different between two of the soil polygons.

MuSym	MuName
6	Pulaski fine sandy loam, 0 to 1 percent slopes, frequently flooded
GrLE	Grainola-Lucien complex, 5 to 12 percent slopes, rocky
ZaHC	Zaneis-Huska complex, 1 to 5 percent slopes
42	Ashport silty clay loam, 0 to 1 percent slopes, occasionally flooded
EasA	Easpor loam, 0 to 1 percent slopes, occasionally flooded
42	Ashport silty clay loam, 0 to 1 percent slopes, occasionally flooded
4	Coyle loam, 3 to 5 percent slopes, eroded

Table 2 - Selected Soils

The seeds selected based on the relative maturity window, ordered by the recommendation rating are shown in Table 3:

SeedID	Seed Name	Suitability Rating
4	33B54	79
7	31R87	77
10	38B85	67
5	32T84	66
6	31N28	65
2	37Y11	64
3	34A15	58

Table 3 - Seed recommendations, no miscellaneous facts

As we can see the two top rated varieties were scored very near each other, within 2 “points” of each other, while the third best variety lags behind by a larger margin. The details for each variety are given in Table 4.

Seed ID	4	7	10
Seed Name	33B54	31R87	38B85
RM	113	120	98
Drought Tol	8	8	7
Root Strength	7	6	6
Stalk Strength	5	6	6
Leaf Blight res	5	5	5
Common Rust res	5	4	4
Stalk Rot res	4	7	4
Ear Rot res	5	5	3
Roundup ok	1	1	1
Liberty ok	0	0	1

Corn borer ok	1	0	0
Rootworm ok	0	0	0

Table 4 - Top three seed details

As is readily apparent, the top two scoring varieties had a better rating for drought tolerance, since drought tolerance is one of the primary categories in the inference engine, it's to be expected that varieties with lower drought tolerance values will score lower without some other condition to increase its suitability rating. The top scoring variety also had an advantage over second place in both root strength (major category), and common rust resistance (a minor category), while the second-best variety had a large advantage with stalk rot resistance. Since that is not a major category in the inference engine rule set, the benefit of the higher score was outweighed by the other disadvantages compared to the top scoring variety.

Run 2 - Some options set

Using the same settings above with the following changes yields a different set of results. By setting “Liberty used” to true and “protect from rootworm” to true, while leaving all other values the same, we generate a new list of suitability ratings shown in Table 5.

Seed ID	Seed Name	Suitability Rating
5	32T84	86
4	33B54	79
7	31R87	77
10	38B85	77
6	31N28	65
2	37Y11	64
3	34A15	58

Table 5 - Ratings with some liberty, rootworm

We see that seed ID 5 – 32T84 has moved from fourth place to first place, and the ratings for the previous top three have become much closer to one another, so much so that there is a tie for 3rd place where there was once a ten point difference. An examination of the seed values in SQL Server illustrates the reasons for this. As seen in Table 6, the new first-place seed has similar ratings for drought tolerance as the second- and third-place values, and a very low rating for root strength, which is a primary category, thus the lower rating in the first run. However, it is the only variety listed among the four that resists the Liberty herbicide while also providing protection against the corn borer beetle. By providing for both of these special requests, and maintaining good values in the primary categories, it successfully supplants the previous recommendations.

Seed ID	5	4	7	10
Seed Name	32T84	33B54	31R87	38B85
RM	115	113	120	98
Drought Tol	8	8	8	7
Root Strength	4	7	6	6
Stalk Strength	6	5	6	6
Leaf Blight res	5	5	5	5
Common Rust res	5	5	4	4
Stalk Rot res	5	4	7	4
Ear Rot res	5	5	5	3
Roundup ok	1	1	1	1
Liberty ok	1	0	0	1
Corn borer ok	1	1	0	0
Rootworm ok	1	0	0	0

Table 6 - Seed Details for top four varieties with options set

CHAPTER VI

SUMMARY

Conclusions

It appears that using decision support systems leads to an effective method to select seed varieties based on the soil type being planted, the seeds characteristics, and user preferences for pest protection and pesticide tolerance. By leveraging the strengths of databases, OGC standards-compliant services, expert system shells, and the C# language, the construction of such a suitable system was illustrated. The challenge of standardizing seed characteristic data and soil data into a useable format was addressed. Similarly, the ability to use a range of different systems to solve problems was also demonstrated. By providing a list of seeds, ranked by suitability, to the end user, the system provides an aid in decision making and planning for the next planting session.

Future/Suggested Work

During the development of this knowledge-based system, several areas for potential work were revealed. The largest area for investigation is in the use and methodology of the expert system. The second area for future work would be in the investigation of the soil data and the geographic operations that can be performed there. Other general items to implement would include support for multiple types of crops,

printable reporting for the end users and an expansion of seed data to support more brands in the standardized seed data format.

A primary item to investigate is the expert system's ability to use different sets of rules, based on different expert opinions. Another item to investigate would be recommendation modification based on a user's preferred amount of risk to maximize yield. A third item for investigation is the inclusion of other sources of data that may affect the field, such as satellite imagery and weather data. Finally, many of the other soil characteristics can be used to enhance the recommendation.

For the soil data and geographic work, a geographic intersection could be performed to eliminate soil types that lie within the extent (bounding box) of the field area, but not within the field boundary itself. Another item of work with this data would be an area calculation for the individual soil polygons, used to weight the results towards the more prominent soil types instead of the most restrictive one.

REFERENCES

- Diluzio, M., J. G. Arnold, and R. Srinivasan. "Integration of SSURGO maps and soil parameters within a geographic information system and nonpoint source pollution model." Journal of Soil and Water Conservation 59 (2004): 123-133.
- ISO. "ISO/IEC 23270:2006(E) Information Technology – programming languages – C#." International Organization for Standardization. 2006. <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=42926>.
- ECMA. "ECMA-334 C# Language Specification." European association for standardizing information and communication systems. 2006. <<http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf>>.
- Egenhofer, M. J.. "Spatial SQL: A Query and Presentation Language." IEEE Transactions on Knowledge and Data Engineering 6 (1994): 86-95.
- Soil Survey Division Staff. "Soil Survey Manual." U.S. Department of Agriculture Handbook 18. Soil Conservation Service, 1993.
- Sprague, R. H., and E. D. Carlson. Building effective decision support systems. 1st ed. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- Wang, W., B. Vincour, and A. Altman. "Plant responses to drought, salinity and extreme temperatures: towards genetic engineering for stress tolerance." Planta 218 (2003): 1-14.

APPENDIX A
CLIPS RULES FILE

```
(deftemplate SOIL_DEF
  (slot mukey
    (type STRING)
    (default ""))
  (slot musym
    (type STRING)
    (default ""))
  (slot muname
    (type STRING)
    (default ""))
  (slot availWater
    (type FLOAT)
    (default 0.0))
  (slot floodFreq
    (type INTEGER)
    (default 0))
  (slot drainageClass
    (type INTEGER)
    (default 0))
  (slot pH
    (type FLOAT)
    (default 7.0))
  (slot organicMatter
    (type FLOAT)
    (default 0.0))
  (slot waterRating
    (type INTEGER)
    (default 50))
)
```

```
(deftemplate SEED_DEF
  (slot seedID
    (type INTEGER)
    (default 1))
  (slot sName
    (type STRING)
    (default ""))
  (slot rm
    (type INTEGER)
    (default 1))
  (slot droughtTol
    (type INTEGER)
    (default 1))
)
```

```

(slot stalkStr
  (type INTEGER)
  (default 1))
(slot rootStr
  (type INTEGER)
  (default 1))
(slot stalkRot
  (type INTEGER)
  (default 1))
(slot leafBlight
  (type INTEGER)
  (default 1))
(slot commonRust
  (type INTEGER)
  (default 1))
(slot earRot
  (type INTEGER)
  (default 1))
(slot roundup
  (type STRING)
  (default "FALSE"))
(slot liberty
  (type STRING)
  (default "FALSE"))
(slot cornBorer
  (type STRING)
  (default "FALSE"))
(slot rootworm
  (type STRING)
  (default "FALSE"))
(slot suitRate
  (type INTEGER)
  (default 50))
)

```

```

;In order to both simplify code and prevent infinite loops, once a
;value has been checked for ratings, it will be set to nil - once nil
;it won't re-activate the rule.

```

```

;Rules that update seed and soil options based on the misc requests
; values that affect the seed's suitability rating based on user
; request misc items can have any salience value higher than the
; value for modifying suitRate based on soil values

```

```

(defrule upSeedCornBorer
  (declare (salience 50))
  (Protect CornBorer)
  ?seedWithCB <- (SEED_DEF (cornBorer "TRUE")
                  (suitRate ?SR))
  =>
  (modify ?seedWithCB(suitRate (+ ?SR 10))
          (cornBorer "nil"))
)
)

```

```

(defrule upSeedRootWorm
  (declare (salience 50))
)

```

```

(Protect RootWorm)
?seedWithRW <- (SEED_DEF (rootworm "TRUE")
                (suitRate ?SR))
=>
(modify ?seedWithRW(suitRate (+ ?SR 10))(rootworm "nil"))

(defrule upSeedRoundup
  (declare (salience 50))
  (Pest Roundup)
  ?seedWithRU <- (SEED_DEF (roundup "TRUE")
                  (suitRate ?SR))
=>
(modify ?seedWithRU(suitRate (+ ?SR 10))(roundup "nil"))

(defrule upSeedLiberty
  (declare (salience 50))
  (Pest Liberty)
  ?seedWithLib <- (SEED_DEF (liberty "TRUE")
                   (suitRate ?SR))
=>
(modify ?seedWithLib(suitRate (+ ?SR 10))(liberty "nil"))

; rules that modify the soil's water rating based on field options
(defrule upSoilTiled
  (declare (salience 50))
  ?field-is-tiled <- (FieldOp Tiled)
  ?soilTiled <- (SOIL_DEF (waterRating ?WR))
=>
(modify ?soilTiled(waterRating (- ?WR 5)))
(retract ?field-is-tiled)
)

(defrule upSoilTerraced
  (declare (salience 50))
  ?field-is-terraced <- (FieldOp Terraced)
  ?soilTerraced <- (SOIL_DEF (waterRating ?WR))
=>
(modify ?soilTerraced(waterRating (+ ?WR 5)))
(retract ?field-is-terraced)
)

;rules for adjusting the various soil water ratings
;these must have higher salience than the next section that adjusts
;seed ratings based ON these values
;first, available water
;modify the water rating based on the available water
; less than 13 is bad, 13-16 is typical range, greater than 16 is
; kind of high
(defrule availWaterLow
  (declare (salience 40))
  ?soil <- (SOIL_DEF (waterRating ?WR)
            (availWater ?aw))
  (test (and (< ?aw 13) (> ?aw 0)))
=>
(modify ?soil(waterRating (- ?WR 20)) (availWater -1.0))

```

```

(defrule availWaterHigh
  (declare (salience 40))
  ?soil <- (SOIL_DEF (waterRating ?WR)
              (availWater ?aw))
  (test (and (> ?aw 16) (> ?aw 0)))
  =>
  (modify ?soil(waterRating (+ ?WR 20))(availWater -1.0)))

;now, flood frequency
;0 = no flooding, 1 = rare, 2 = occasional, 3 = frequent.
;treat 1 as the non-modifier case
(defrule floodLow
  (declare (salience 40))
  ?soil <- (SOIL_DEF (waterRating ?WR)
              (floodFreq ?ff))
  (test (eq ?ff 0))
  =>
  (modify ?soil(waterRating (- ?WR 10)) (floodFreq -1)))

(defrule floodOcc
  (declare (salience 40))
  ?soil <- (SOIL_DEF (waterRating ?WR)
              (floodFreq ?ff))
  (test (eq ?ff 2))
  =>
  (modify ?soil(waterRating (+ ?WR 5)) (floodFreq -1)))

(defrule floodOften
  (declare (salience 40))
  ?soil <- (SOIL_DEF (waterRating ?WR)
              (floodFreq ?ff))
  (test (eq ?ff 3))
  =>
  (modify ?soil(waterRating (+ ?WR 10)) (floodFreq -1)))

;drainageclass
;0,1 - excessive drainage through 6 - very poor drainage, value
; of 4 will not modify the rating
(defrule drainExcess
  (declare (salience 40))
  ?soil <- (SOIL_DEF (waterRating ?WR)
              (drainageClass ?dc))
  (test (and (<= ?dc 1) (>= ?dc 0)))
  =>
  (modify ?soil(waterRating (- ?WR 10)) (drainageClass -1)))

(defrule drainSomeExcess
  (declare (salience 40))
  ?soil <- (SOIL_DEF (waterRating ?WR)
              (drainageClass ?dc))
  (test (eq ?dc 2))
  =>
  (modify ?soil(waterRating (- ?WR 5))(drainageClass -1)))

(defrule drainModerate
  (declare (salience 40))

```

```

?soil <- (SOIL_DEF (waterRating ?WR)
              (drainageClass ?dc))
(test (eq ?dc 3))
=>
(modify ?soil(waterRating (- ?WR 3))(drainageClass -1))

(defrule drainSomePoor
  (declare (salience 40))
  ?soil <- (SOIL_DEF (waterRating ?WR)
              (drainageClass ?dc))
  (test (eq ?dc 5))
  =>
  (modify ?soil(waterRating (+ ?WR 5))(drainageClass -1))

(defrule drainPoor
  (declare (salience 40))
  ?soil <- (SOIL_DEF (waterRating ?WR)
              (drainageClass ?dc))
  (test (eq ?dc 6))
  =>
  (modify ?soil(waterRating (+ ?WR 10))(drainageClass -1))

;Find the WORST soil - this must occur AFTER the soils have been
;classified above
(defrule worstWaterRating
  (declare (salience 35))
  (SOIL_DEF (waterRating ?wr))
  (not (SOIL_DEF (waterRating ?b&:(< ?b ?wr))))
  =>
  (assert (worst-soil-rating ?wr))

;Now seed manipulation based on the soil water rating - these must have
; a salience value lower than all other rules
;Note - the VAST majority of ratings values fall in the range of 4-8
; inclusive, so the logic here varies around that range. A value of 0
; means that a characteristic has not been satisfactorily measured by
; the seed company

; drought tol
(defrule modifyDroughtTolFour
  (declare (salience 30))
  ?seed <- (SEED_DEF (droughtTol ?dt)
              (suitRate ?sr))
  ?worstWR <- (worst-soil-rating ?wwr)
  (test (and (<= ?dt 4) (> ?dt 0)))
  =>
  (modify ?seed(suitRate (+ ?sr (- ?wwr 70)))(droughtTol -1))

(defrule modifyDroughtTolFive
  (declare (salience 30))
  ?seed <- (SEED_DEF (droughtTol ?dt)
              (suitRate ?sr))
  ?worstWR <- (worst-soil-rating ?wwr)
  (test (eq ?dt 5))
  =>
  (modify ?seed(suitRate (+ ?sr (- ?wwr 65)))(droughtTol -1))

```

```

(defrule modifyDroughtTolSix
  (declare (salience 30))
  ?seed <- (SEED_DEF (droughtTol ?dt)
             (suitRate ?sr))
  ?worstWR <- (worst-soil-rating ?wwr)
  (test (eq ?dt 6))
  =>
  (modify ?seed(suitRate (+ ?sr (- ?wwr 60)))(droughtTol -1)))

(defrule modifyDroughtTolSeven
  (declare (salience 30))
  ?seed <- (SEED_DEF (droughtTol ?dt)
             (suitRate ?sr))
  ?worstWR <- (worst-soil-rating ?wwr)
  (test (eq ?dt 7))
  =>
  (modify ?seed(suitRate (+ ?sr (- ?wwr 55)))(droughtTol -1)))

(defrule modifyDroughtTolEight
  (declare (salience 30))
  ?seed <- (SEED_DEF (droughtTol ?dt)
             (suitRate ?sr))
  ?worstWR <- (worst-soil-rating ?wwr)
  (test (>= ?dt 8))
  =>
  (modify ?seed(suitRate (+ ?sr (- ?wwr 50)))(droughtTol -1)))

;Root strength
(defrule modifyRootStrFour
  (declare (salience 30))
  ?seed <- (SEED_DEF (rootStr ?rs)
             (suitRate ?sr))
  ?worstWR <- (worst-soil-rating ?wwr)
  (test (and (<= ?rs 4) (> ?rs 0)))
  =>
  (modify ?seed(suitRate (- ?sr (- ?wwr 40)))(rootStr -1)))

(defrule modifyRootStrFive
  (declare (salience 30))
  ?seed <- (SEED_DEF (rootStr ?rs)
             (suitRate ?sr))
  ?worstWR <- (worst-soil-rating ?wwr)
  (test (eq ?rs 5))
  =>
  (modify ?seed(suitRate (- ?sr (- ?wwr 45)))(rootStr -1)))

(defrule modifyRootStrSix
  (declare (salience 30))
  ?seed <- (SEED_DEF (rootStr ?rs)
             (suitRate ?sr))
  ?worstWR <- (worst-soil-rating ?wwr)
  (test (eq ?rs 6))
  =>
  (modify ?seed(suitRate (- ?sr (- ?wwr 50)))(rootStr -1)))

(defrule modifyRootStrSeven
  (declare (salience 30))

```



```

?seed <- (SEED_DEF (rootStr ?rs)
            (suitRate ?sr))
?worstWR <- (worst-soil-rating ?wwr)
(test (eq ?rs 7))
=>
(modify ?seed(suitRate (- ?sr (- ?wwr 55)))(rootStr -1)))

(defrule modifyRootStrEight
  (declare (salience 30))
  ?seed <- (SEED_DEF (rootStr ?rs)
            (suitRate ?sr))
  ?worstWR <- (worst-soil-rating ?wwr)
  (test (>= ?rs 8))
=>
  (modify ?seed(suitRate (- ?sr (- ?wwr 60)))(rootStr -1)))

;Bonuses
; Seeds contain a lot of resistances that aren't tied directly to soil
; types In order to recognize those values, we add unused resistances
; to the overall seed rating here, this will also help break ties if
; several seeds with similar drought tolerance and root strength are
; present in the facts list.
(defrule upStalkStr
  (declare (salience 20))
  ?seed <- (SEED_DEF (stalkStr ?ss) (suitRate ?sr))
  (test (>= ?ss 0))
=>
  (modify ?seed(suitRate (+ ?sr ?ss))(stalkStr -1)))

(defrule upStalkRot
  (declare (salience 20))
  ?seed <- (SEED_DEF (stalkRot ?srr) (suitRate ?sr))
  (test (>= ?srr 0))
=>
  (modify ?seed(suitRate (+ ?sr ?srr))(stalkRot -1)))

(defrule upLeafBlight
  (declare (salience 20))
  ?seed <- (SEED_DEF (leafBlight ?lbr) (suitRate ?sr))
  (test (>= ?lbr 0))
=>
  (modify ?seed(suitRate (+ ?sr ?lbr))(leafBlight -1)))

(defrule upCommonRust
  (declare (salience 20))
  ?seed <- (SEED_DEF (commonRust ?crr) (suitRate ?sr))
  (test (>= ?crr 0))
=>
  (modify ?seed(suitRate (+ ?sr ?crr))(commonRust -1)))

(defrule upEarRot
  (declare (salience 20))
  ?seed <- (SEED_DEF (earRot ?err) (suitRate ?sr))
  (test (>= ?err 0))
=>
  (modify ?seed(suitRate (+ ?sr ?err))(earRot -1)))

```

```
;special for clipsNET
;a final rule with lowest salience to fire when everything else has
;gone through
(defrule allDoneRule
  (declare (salience 5))
  (test (eq 0 0))
  =>
  (GrabResults))
```

APPENDIX B

CUIManager Class

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using ClipsNET;
using System.Windows.Forms;

namespace SoilAnswers
{
    //for clips event handling
    public delegate void ClipsFunction(IntPtr p);

    //for notification of successful completion
    public delegate void ResultsReadyDelegate(DataView dv);

    public class CUIManager
    {
        public event ResultsReadyDelegate ResultsReady;

        #region geography
        //Data bounding box for Payne County (our dataset)
        public const double BB_MIN_LON = -97.3584716264595;
        public const double BB_MAX_LON = -96.6121166365871;
        public const double BB_MIN_LAT = 35.9315811070456;
        public const double BB_MAX_LAT = 36.2549522766129;

        private double minLon;
        public double MinLon
        {
            get { return minLon; }
            set { minLon = value; }
        }

        private double maxLon;
        public double MaxLon
        {
            get { return maxLon; }
            set { maxLon = value; }
        }

        private double minLat;
        public double MinLat
        {
            get { return minLat; }
            set { minLat = value; }
        }
    }
}
```

```

}

private double maxLat;
public double MaxLat
{
    get { return maxLat; }
    set { maxLat = value; }
}

private DataTable dtSelectedSoils;
public DataTable DtSelectedSoils
{
    get { return dtSelectedSoils; }
    set { dtSelectedSoils = value; }
}

#endregion

#region seed options
private int relMatMin;
public int RelMatMin
{
    get { return relMatMin; }
    set { relMatMin = value; }
}

private int relMatMax;
public int RelMatMax
{
    get { return relMatMax; }
    set { relMatMax = value; }
}

private bool protectRootworm;
public bool ProtectRootworm
{
    get { return protectRootworm; }
    set { protectRootworm = value; }
}

private bool protectCornBorer;
public bool ProtectCornBorer
{
    get { return protectCornBorer; }
    set { protectCornBorer = value; }
}
#endregion

#region field options
private bool isTiled;
public bool IsTiled
{
    get { return isTiled; }
    set { isTiled = value; }
}

private bool isTerraced;

```

```

public bool IsTerraced
{
    get { return isTerraced; }
    set { isTerraced = value; }
}

private bool useRoundup;
public bool UseRoundup
{
    get { return useRoundup; }
    set { useRoundup = value; }
}

private bool useLiberty;
public bool UseLiberty
{
    get { return useLiberty; }
    set { useLiberty = value; }
}
#endregion

#region Clips objects and fact lists
private ClipsEngine engine;

private const string RULE_FILE = @"rules.clp";

private List<CSeedFact> seedFacts;
public List<CSeedFact> SeedFacts
{
    get { return seedFacts; }
}

private List<CSoilFact> soilFacts;
public List<CSoilFact> SoilFacts
{
    get { return soilFacts; }
}

private List<CMiscFact> miscFacts;
public List<CMiscFact> MiscFacts
{
    get { return miscFacts; }
    set { miscFacts = value; }
}

//templates for soils and seeds
private Deftemplate seedTemplate;
public Deftemplate SeedTemplate
{
    get { return seedTemplate; }
}

private Deftemplate soilTemplate;
public Deftemplate SoilTemplate
{
    get { return soilTemplate; }
}

```

```

#endregion

public CUIManager()
{
    //load some default values to help the user
    this.isTerraced = false;
    this.isTiled = false;
    this.protectCornBorer = false;
    this.protectRootworm = false;
    this.useRoundup = false;
    this.useLiberty = false;

    this.relMatMin = 80;
    this.relMatMax = 130;

    //north western quad-sphere
    this.minLon = -180.0;
    this.maxLon = 0.0;
    this.minLat = 0;
    this.maxLat = 180;

    this.dtSelectedSoils = new DataTable();
    dtSelectedSoils.Columns.Add("MUSym",
        Type.GetType("System.String"));
    dtSelectedSoils.Columns.Add("MUKey",
        Type.GetType("System.String"));

    this.seedFacts = new List<CSeedFact>();
    this.soilFacts = new List<CSoilFact>();
    this.miscFacts = new List<CMiscFact>();
    initClips();
}

private void initClips()
{
    this.engine = new ClipsEngine();

    //define C# functions to execute with CLIPS
    this.engine.DefineFunction(new
        ClipsFunction(this.GrabResults));

    //load the rules file
    this.engine.Load(RULE_FILE);
    this.engine.Reset();

    //load some templates to make life easier
    soilTemplate = engine.Deftemplates["SOIL_DEF"];
    seedTemplate = engine.Deftemplates["SEED_DEF"];
}

#region retract/assert facts
public void RetractSoilFacts()
{
    //retract each fact
    foreach (CSoilFact csfact in this.soilFacts)
    {

```

```

        engine.Retract(csfact);
    }

    //now clear the fact list
    this.soilFacts.Clear();
}

public void AssertSoilFacts()
{
    //assert the new facts into clips
    foreach (CSoilFact csfact in this.soilFacts)
    {
        csfact.AssignSlotValues();
        engine.Assert(csfact);
    }
}

public void RetractSeedFacts()
{
    //remove from clips
    foreach (CSeedFact csfact in this.seedFacts)
    {
        engine.Retract(csfact);
    }

    //clear the internal list
    this.seedFacts.Clear();
}

public void AssertSeedFacts()
{
    //assert the new facts into clips
    foreach (CSeedFact csfact in this.seedFacts)
    {
        csfact.AssignSlotValues();
        engine.Assert(csfact);
    }
}

public void RetractMiscFacts()
{
    //remove from clips
    foreach (CMiscFact msfact in this.miscFacts)
    {
        if (engine.FactList.Contains(msfact))
        {
            engine.Retract(msfact);
        }
    }

    //clear the internal list
    this.miscFacts.Clear();
}

public void AssertMiscFacts()
{
    //assert the new facts into clips

```

```

        foreach (CMiscFact msfact in this.miscFacts)
        {
            engine.Assert(msfact);
        }
    }
#endregion

internal void RunEngine()
{
    this.engine.Run();
}

private void GrabResults(IntPtr p)
{
    //Clips has finished. gather what we need.
    DataTable dtResults = new DataTable();
    dtResults.Columns.Add("seedID",
        Type.GetType("System.String"));
    dtResults.Columns.Add("name",
        Type.GetType("System.String"));
    dtResults.Columns.Add("Rating",
        Type.GetType("System.Int32"));

    //since our clips program will modify the facts, we can't
    // use the objects we made earlier.
    //examine the facts list for new facts
    foreach (Fact fct in this.engine.FactList)
    {
        if (fct.Text.Contains("SEED_DEF"))
        {
            try
            {
                //we can use this, it's a seed fact
                DataRow dr = dtResults.NewRow();
                //parse the string
                string fctStr = fct.Text;
                int idIdx = fctStr.IndexOf("(seedID ") + 8;
                string idStr = fctStr.Substring(idIdx,
                    fctStr.IndexOf(")", idIdx) - idIdx);

                int nmIdx = fctStr.IndexOf("(sName ") + 7;
                string nmStr = fctStr.Substring(nmIdx,
                    fctStr.IndexOf(")", nmIdx) -
                    nmIdx).Replace("\\", "");

                int srIdx = fctStr.IndexOf("(suitRate ") + 10;
                string srStr = fctStr.Substring(srIdx,
                    fctStr.IndexOf(")", srIdx) - srIdx);

                dr["seedID"] = Convert.ToInt32(idStr);
                dr["name"] = nmStr;
                dr["Rating"] = Convert.ToInt32(srStr);
                dtResults.Rows.Add(dr);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error parsing facts: " +

```



```

        ex.Message + "\r\n" + ex.StackTrace);
    }
}
dtResults.AcceptChanges();

//sort the datatable's defaultview by rating, descending
dtResults.DefaultView.Sort = "Rating DESC";

//shut down the clips engine
this.engine.RetractAll();
this.engine.Dispose();
GC.Collect();

this.initClips(); //in case we want to run it again

//send out an event for displaying results
if (this.ResultsReady != null)
{
    ResultsReady(dtResults.DefaultView);
}
}
}
}

```

VITA

Phillip Andrew Martin

Candidate for the Degree of

Master of Science

Thesis: **DECISION SUPPORT SYSTEM FOR SEED SELECTION USING
SPATIALLY-REFERENCED SOIL DATA**

Major Field: Computer Science

Biographical:

Personal Data: Born in Springfield, Missouri, on August 24th, 1978, the son of Steve and Karen Martin.

Education: Graduated from Kellyville High School, Kellyville, OK in May 1996; received Bachelor of Science degree in Computer Science from Oklahoma State University, Stillwater, Oklahoma in December 2000. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in July 2009.

Experience: Employed as a software engineer at Worldcom in Tulsa, Oklahoma from December 2000 to April 2002. Employed as a teaching assistant at Oklahoma State University from August 2002 to May 2003. Employed as a software engineer at SST Software in Stillwater, Oklahoma from May 2003 to present.

Professional Memberships: None

Name: Phillip Andrew Martin

Date of Degree: July, 2009

Institution: Oklahoma State University

Location: Stillwater, Oklahoma

Title of Study: **DECISION SUPPORT SYSTEM FOR SEED SELECTION
USING SPATIALLY-REFERENCED SOIL DATA**

Pages in Study: 49

Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study: The purpose of this study was to attempt to recommend specific varieties of seeds to an end user based upon the soils in the area being planted, as well as some user-defined options for field logistics. The system created involved the use of a WMS service, a WFS service, an RDBMS, the CLIPS expert system shell, and the C# programming language.

Findings and Conclusions: The conglomeration of the independent pieces was found to successfully create a system capable of recommending seed varieties based on a suitability rating. This suitability rating was created via examination of characteristics of the seeds offered, the soils being planted, and the individual user's own preferences for protection from pests and tolerance of pesticides.

ADVISER'S APPROVAL: _____