REGEN AGENT:

A MODULAR AND GENERALIZED FOREST REGENERATION AGENT

by

DAVID AGUSTIN BOUCUGNANI

(Under the direction of Donald Nute)

ABSTRACT

This thesis describes the implementation of two software applications: REGEN Agent and REGEN Agent for Excel. REGEN Agent is an expert system and modeling application providing a mechanism for forest regeneration prediction based on the Loftis regeneration model. The Loftis model is both a stochastic and a competitive model that utilizes expert knowledge encoded in a knowledge base and applies this knowledge utilizing an underlying engine. REGEN Agent for Excel is a composition of both REGEN Agent and a client application written in Visual Basic for Applications and implemented in Microsoft Excel. This end-user application makes the REGEN model accessible to end-users in both the expert and management domains. Theoretical foundations, programming methodology and a sample application are provided in the thesis document, which is intended and targeted as a general resource for all interested users.

INDEX WORDS: Forest regeneration, Ecosystem simulation, Competitive model, Prolog-VBA interface

REGEN AGENT:

A MODULAR AND GENERALIZED FOREST REGENERATION AGENT

by

David Agustin Boucugnani

B.S., University of Georgia, 2000

A Thesis Submitted to the Graduate Faculty

of the University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2005

REGEN AGENT:

A MODULAR AND GENERALIZED FOREST REGENERATION AGENT

by

DAVID AGUSTIN BOUCUGNANI

Major Professor: Donald Nute

Committee: Walter D. Potter
                     Khaled Rasheed

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2005

DEDICATION

There is no greater enterprise, nor no greater responsibility, than having, caring for and loving children. Alexis and Sydney Boucugnani are the lights of my life and have been with me throughout my education, endeavors, successes and failures in my quest to understand the fascinating adventure that is adult life. I cannot thank them enough for their support and unconditional love. You have both helped me more that you may ever realize.

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The purpose of this thesis is to document the design and implementation of an agent, termed

REGEN Agent, for modeling the process of regeneration in silvicultural systems. By utilizing a

concept of "intuition encoding", experts are able to develop a stochastic model by which classes

of individuals (in this case, species and size categories of trees) compete for representation in a

particular area. Additionally, this thesis documents the design and construction of a graphical

user interface (GUI) to REGEN Agent, titled "REGEN for Excel", which when combined with

the Microsoft Excel spreadsheet package yields a familiar interface for most end users. In this

chapter, a summary of REGEN Agent is provided along with a history of the original

implementation and further extensions/modifications. In the second chapter, the underlying

ecological/silvicultural theory behind the model is presented, along with a discussion of the

artificial intelligence methodologies utilized. Chapter 3 details the implementation of the model

described in Chapter 2 including software engineering considerations while Chapter 4 details the

user interface design and components of REGEN for Excel. Finally, a sample application is

offered, exhibiting the end-use of the REGEN for Excel Knowledge Editor, Stand Editor and

Report Generator along with a critical review of the project to-date.

1.1 THE REGEN AGENT PROJECT: A SUMMARY

The REGEN model is applied for the most part to silvicultural systems to describe post-harvest or post-disturbance performance of a given stand of trees. The underlying abstract model utilizes a ranking scheme based on species/size classes as well as stochastic functions to determine both new seedling establishment and probability of stump sprouting (i.e., the ability of certain species to grow from cut or partially destroyed mature individuals). The model is applied to an inventory of potential regeneration sources in the existing stand coupled with a knowledge base of both rankings and stochastic coefficients for each regeneration source to provide a projection of what the species composition of dominant and/or codominant individuals is likely to be in the new (post-disturbance) stand at the time of crown closure (when competition essentially ceases due to lack of sunlight in the understory). Given the model's inherent random nature, multiple simulations can provide interesting statistics such as mean, standard deviation and distribution metrics.

1.2 A BRIEF HISTORY OF THE REGEN PROJECT

Recently, the Loftis regeneration model, REGEN (1989, 1990) was implemented using a decision support system authoring tool entitled AppBuilder for DSSTools (Kim et al. 2000). While this tool's utility for developing decision support systems in general is clear, problems with extensibility and modularity between the core engine and interface indicated a need for an architectural rework. The general consensus among the interested parties was that the REGEN model should be implemented as an agent containing no human interface code and with a

separation between declarative and procedural knowledge. Then the agent could serve as a "plug-in" for any number of custom human-computer interfaces or as a module in larger forestry applications such as NED-2 (Nute et al. 2004).

REGEN has been extended to provide for a 2-level hierarchical arrangement of ecological or management units (e.g., Southern Appalachians and subxeric oak, respectively) so that unit-specific information can be provided for in a separate knowledge base. Indeed, with the disambiguation of the declarative knowledge base from the procedural core engine, any number of custom knowledge bases may be constructed and selected during stand composition definition. In addition, probability metrics for new seedling establishment have been extended to include logistic probability estimators. A provision for an alternate plot size (e.g., $1/250^{th}$ acre) has also been included.

On the software side, a modular agent architecture has been employed that can understand and utilize a separate knowledge and rule base, allowing for customization of both the model and domain without significant programming effort. The agent is wrapped into a module usable by any number of custom interface schemes (e.g., REGEN for Excel). Indeed, this modularity is conducive for inclusion of the model in any decision model where post-disturbance species composition is an issue. Other extensions include facilities for batching multiple runs with calculation of various statistical metrics and a new "root sucker" or underground stem regeneration source.

Development of the agent has allowed for a new standalone software package implemented in Microsoft Excel. The driver for this interface choice was the fact that most potential end-users should be familiar with spreadsheet packages in general and the

overwhelming popularity of the Microsoft Excel spreadsheet package in particular. Microsoft

Excel provides familiar data entry, tracking and charting, and report generation facilities.

Output and summary tables are displayed as native Excel sheet objects, but can be exported to a

number of other formats such as HTML.

CHAPTER 2

DOMAIN ANALYSIS

2.1   SILVICULTRAL FOUNDATIONS

REGEN is a predictive model based on the theory that species composition of a new stand of trees after a significant disturbance is dependent in large part upon species composition before the event occurs. In particular, existing regeneration sources such as persistent seedlings, saplings and root systems and new, incoming regeneration sources such as windblown and animal-carried seeds define the species composition of the new stand (Egler 1954, Noble and Slatyer 1980). Because REGEN is a competitive model regeneration sources compete with each other using a consistent method of comparison, namely, expert ranking of individual species/size couplets. Those individuals that "win" in the competition for dominance appear in the regenerated stand, while those individuals that "lose" lack representation.

The theory and the model in general assume consistent species regeneration strategies within a particular situation. For example, certain conifer species have adapted to regeneration after fire (a common natural disturbance) by evolving *serotinous cones*, which protect their contained seeds from fire and simultaneously disseminate seeds after the fire, when most potential competition has been eliminated. Other species have adapted to survive disturbances by evolving mechanisms to avoid fatal damage to the existing individual (e.g., high branches and leaves to avoid overgrazing and understory fire.)   Additional mechanisms include the

comparative rate at which newly-established seedlings grow (e.g., the fast-growing *Liliodendron tulipifera* or Tulip poplar) and the propensity for certain species, like *Quercus* spp. or oak, to sprout from stumps or underground stems or root systems. While the above illustrate only a few of the many adaptations to disturbance, the implication is that each species has a preferred regeneration strategy to match a given ecosystem and disturbance situation, allowing for the application of a singular ranking matching the preferred regeneration mechanism. Without this rule, application of the theory would be complicated by multiple rankings given to the same individual based on competing regeneration strategies.

Relatively few regeneration models have been based on the Egler's initial floristics concept, even though the general structure of the theory has been shown consistent with real-world observation of secondary succession (Oliver and Larson 1996; Drury and Nisbet 1973; Kimmins 1996; Peet and Christensen 1980.) Most of the models sharing the initial domain of the REGEN model (Sander et al. 1984; Dey 1991; Loftis 1988 1990), namely upland oak forests or upland forests where oak is a primary regeneration component are based on both the notion that oak tends to persist through disturbance as advance reproduction (seedlings and saplings) and through living root systems of harvested trees and that size of advance growth affects post-disturbance performance. Unlike REGEN, these models do not estimate the total species composition of the stand in question but rather predict only the expected amount of oak.

Ferguson et al. (1986) developed a comprehensive regeneration prediction model for the relatively species-rich mixed conifer stands in northern Idaho. Using empirical data gathered from a variety of silvicultural operations, they developed regression equations to impute and grow advance reproduction, logistic equations to estimate probabilities of new seedling

establishment, and regression equations to grow the seedlings. The resulting tree list is then used as input to a growth simulator for further projection of stand development.

Waldrop et al. (1986) implemented a version of the Shugart and West (1980) FORET model to predict species composition in stands that develop following clearcutting on the Cumberland Plateau. While the model provided reasonable predictions of the species occurrence, predictions of relative abundance were problematic. In particular, the model significantly underestimated the abundance of pioneer species such as *Liriodendron tulipifera* (Tulip poplar).

2.2   COMPARISON TO OTHER IMPLEMENTED REGENERATION MODELS

While there are many theoretical models for forest regeneration, most of them have either not been implemented as computer software or have been implemented as part of holistic forest dynamics simulators such as SORTIE (Deutschman et al. 1997), SILVAH (Ernst 1992), or FVS (Dixon 2003). Many of these models are based on the theory of gap dynamics (Shugart 1984), which details a regeneration process whereby the death of established individuals in a stand releases space and resources for colonization of the new-formed "gap" by regenerants, after which the process of succession ensues and the gap is filled. Most of these models are designed to handle natural disturbance events, which tend to be small-scale or specific, such as in the case of catastrophic fire. The Loftis REGEN model attempts to generalize gap dynamics to more effectively model larger disturbance events including forest management techniques (e.g., clearcutting). While there may be other stand-alone regeneration simulation programs implemented at this time, let us consider ACORn (Dey et al. 1996b), FOREGEN (Solomon and

Leak, 2002), and BLOWDOWN (Peterson unpublished) for comparative analysis to REGEN Agent.

ACORn is a predictive regeneration software package modeling a very narrow domain (important hardwood species in the Missouri Ozark Highlands). ACORn is similar to REGEN in that it is not intended to model gap dynamics, rather, it considers regeneration shortly after complete overstory removal (i.e., forest management and harvest practices) by processing an inventory and comparing against empirical data. However, ACORn cannot be applied to stands containing dominant species not included in the simulator's hard coded database, whereas the generalization provided by the REGEN knowledge structure enables extensibility of REGEN to many management domains. Additionally, stump-sprouting is not considered in non-oak species and certain aggressive species that can appear in Missouri Ozark Highland stands are not included. However, ACORn does take into account site-specific variables such topography due to the granularity of empirical data upon which the model is based, which, in well-defined domains, can provide a more reliable projection. For management purposes, ACORn effectively estimates probable stocking of desirable hardwood species such as oaks and can provide managers in the Missouri Ozark Highlands with crucial what-if analysis capability. Further information on the theory and methodology of ACORn can be obtained from Dey (1991) and Dey et al. (1996a).

FOREGEN is a more generalized regeneration model targeting northern hardwood stands in general and is capable of modeling events ranging from clearcuts to single-tree gaps. It makes heavy use of stochastic methods similar to Monte Carlo and hidden Markov chains in predicting success of regenerants three years from a disturbance event. Modeling specificity is high. For example, variables such as average wind velocity and seed fall rates are included when

considering seed dispersal within the area to be regenerated. Stand-level inventory post-regeneration of both viable and non-viable individuals is produced from a regeneration probability matrix formed by applying empirical, species-specific probabilities of germination and survivorship to the initial regeneration inventory. Because shade and resource tolerance is considered, regeneration in small gaps can be modeled. Competitive dynamics between species in the initial regeneration inventory do not seem to be considered, perhaps due to the 3-year modeling timespan of FOREGEN. REGEN, on the other hand, is primarily a competitive model with some recourse to regenerant establishment probabilities and again is more generalized due to the separation of declarative from procedural knowledge.

BLOWDOWN is an unpublished application modeling disturbance and post-disturbance dynamics by simulating the disturbance itself (i.e., gap formation from events such as wind damage) and by considering seedling dispersal, establishment, early growth and mortality. BLOWDOWN does not model growth past the sapling stage, nor does it consider catastrophic disturbances such as clearcuts or other complete stand elimination events. It is not necessary to model the actual disturbance dynamics in REGEN because of the assumption that REGEN will be applied to such major disturbance events, although provision for plot-level mortality does provide some facility for disturbance dynamics. BLOWDOWN, like REGEN, seems to separate procedural from declarative knowledge in that each time the simulation is run, specific species data ranging from abundance to germination rates must be provided.

CHAPTER 3

REGEN AGENT

Knowledge Editor
- generates the various species knowledge base (KB) files (.rkb) used by both the Agent and Client
- accesses and modifies the REGEN registry, indicating available KBs
- allows for importation and deletion of information on the species and KB levels

Management Unit KB
- encodes species knowledge entered with the Knowledge Editor (e.g., rankings)
- provides for knowledge base identification
- incorporates species name/code information internally for KB specific representation

Read

Write

Consult*

* multiple management knowledge bases may be loaded

Blackboard
- provides working space for REGEN Agent (i.e., temporary knowledge)
- repository of post-simulation data for access by the Client
- attached to the REGEN Agent

Assert

Pass String

Call/Retract

Assert

Client
- provides stand information by direct assertion of facts to blackboard
- allows various management options by direct manipulation of blackboard
- activates REGEN Agent by calling appropriate predicate (function)
- retrieves post-simulation information
- displays or converts data for end-use

Call

REGEN Agent
- takes knowledge from the Client and various Management Unit KBs and performs the regeneration simulation
- protects basal knowledge for multi-run simulations
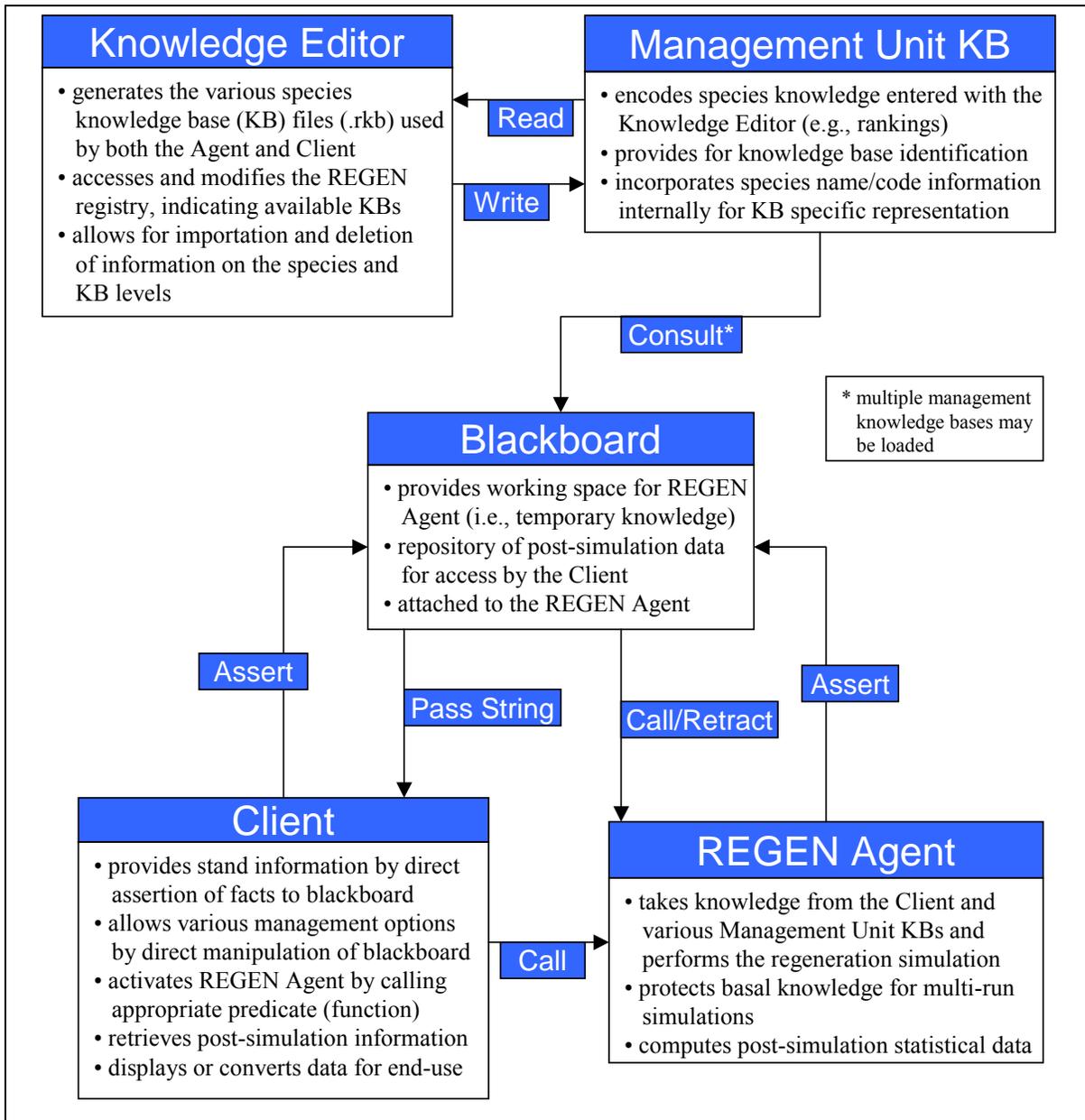- computes post-simulation statistical data

Figure 3.1: Overview of the REGEN Agent Software Architecture

3.1   THE ABSTRACT REGEN MODEL

This section summarizes the Loftis regeneration model (REGEN) as it is implemented in the REGEN Agent architecture. While the REGEN model is intended to estimate post-disturbance species composition, it is only appropriate in the case of a "heavy" disturbance event, either artificial (such as clearcutting) or natural (as in the case of fire). For the purposes of the model, a "heavy" disturbance event is defined as follows: 1) basal area is less than or equal to 50 square feet per acre for large events or, 2) opening size is greater than 0.25 acres and the basal area within the opening is between 0 and 5 feet per acre for localized events. In measuring basal area, only those trees that have achieved 4.5 feet in height are considered.

A key component of REGEN is how pre-disturbance floristic composition contributes to post-disturbance species composition in the regenerated stand. Because the initial load of regeneration propagules directly influences the collection of dominant species in the stand after any heavy disturbance, the number, size, and species of those potential regeneration propagules must be collected in a stand inventory. REGEN calls for a number of plots, either $1/100^{th}$ or $1/250^{th}$ of an acre whereby a given individual in a plot is expanded to a per-acre representation. For a given doublet of species and size, a competitive ranking score (integer) is assigned, whereby competitiveness is directly proportional to the score. Individuals in an inventory are assigned to one of the following size classes: germinant (small seedling), small (≤ 2 feet in height), medium (> 2 feet and ≤ 4 feet in height), large (> 4 feet and ≤ 1.5 inches diameter at breast height) and potential stump sprouts (> 1.5 inches diameter at breast height).

Certain species' seedlings are superior sources of regeneration when compared with other species. For instance, in the central hardwood forest of the United States, yellow poplar, certain

11

birches and to some extent white ash, basswood, and black cherry are characterized by high

seedling densities and rapid juvenile growth (Loftis 1989). Thus, these seedlings become highly

competitive upon sufficient local disturbance and hence are accounted for in the model. Each

"species regenerating from seedlings" is encoded by a *<species,probability,number,rank>*

attribute quadruple, with *probability* being either a constant probability value (see Equation 1) or

a set of coefficients for a logistic establishment function (see Equation 2), *number* indicating the

count of individuals to add to each plot, and *rank* being the rank at which to add each individual.

As of the date of this thesis, variable(s) for the logistic seedling establishment function have not

been identified; however, a placeholder exists in the framework for its later inclusion. Similar to

"species regenerating from seedlings", certain species regenerate from "root suckers" or

underground stems. These are added in much the same way as "species regenerating from

seedlings" except that addition to a given plot is dependent upon a representative of the species

being present on the stand instead of simply adding individuals to each plot. Again, space is

provided for the coefficients for a logistic establishment function, which has yet to be

empirically ascertained.

Particular species have the ability to sprout from stumps and produce competitive

individuals in regenerated stands. The constant and logistic sprouting parameters are defined as

<species,parameters(s)> doubles and combinations, respectively. Thus, each species that is able

to sprout is determined by the preceding attribute sets and the propensity for a particular species

to sprout is influenced by first applying the logistic sprouting probability (Equation 2). If no

logistic sprouting parameters exist, then constant sprouting probability (Equation 1) is employed.

$$1: R \geq P$$
$$0: \text{otherwise}$$

whereby R is a random real variable between 0 and 1 and P is the constant probability parameter

Equation 3.1: Binary Constant Probability Function

$$1: R \geq e^{(P0 + P1*V1 + \dots + PN*VN)}/(1 + e^{(P0 + P1*V1)})$$
$$0: \text{otherwise}$$

whereby R is a random real variable between 0 and 1 and P0 is the independent coefficient and PN is a variable dependent coefficient and VN is the variable for a particular coefficient PN

Equation 3.2: Binary Logistic Probability Function

Once the rankings are defined for species/size class individuals, a plot-wise stand composition is available, and all special cases (i.e., stump sprouts, species regenerating from seedlings and root suckers) are resolved, selection via the competition framework commences. Selection for the plot begins by selecting the top X individuals, where X is determined by plot size. A loop is established whereby individuals of rank *1* are selected, and if there are more individuals to be selected, the selection rank is incremented.

The basal modeling situation occurs when no stump sprouts exist in the given plot. In this situation, comparing the competitive ranking scores derived from the *<species,size,ranking>* triples of each individual in the plot and choosing those with the highest-ranking score first provides this list. If more than one individual can be selected for last place, logic (tiebreaker logic) is needed to handle the collision. In this case, each individual is selected proportionally (e.g., if there are five individuals in last place, each one will be selected as 1/5th of an individual.) This fragmentation is resolved upon expansion of the regenerated plot to a per-acre basis.

If stump sprouts exist in a given plot, a different strategy is employed given the enhanced competitiveness of stump sprouts. Each stump sprout in the given plot is treated as multiple individuals determined by the number of other stump sprouts in the plot. For example, in Southern Appalachian hardwood forests, the current model calls for the following selection algorithm for 1/100th acre plots: if *one* stump sprout is present at a particular rank and no other

selections have been made, select this stump sprout and the top *three* other individuals for a total

of four actual selected individuals. If *two* stump sprouts exist at a particular rank with no prior

selections, select both sprouts and one other individual for a total of three selected individuals.

Finally, if three or more stump sprouts exist at a given rank, take the top two, submitting the

remainder to the tie-breaker logic. For smaller plot sizes, the algorithm should change to reflect

a decrease in competitive space.

Once each plot has been regenerated, the results for each plot are expanded to a per-acre

basis (dependent on the plot size). All regenerated individuals from each expanded plot are

added together, resulting in the expected species composition of the stand at age 10 years. At

this point, it is possible to estimate diameter via a regression relationship defined in the

knowledge base and to provide input to a small tree model such as the Forest Vegetation

Simulator (FVS). Likewise, the regenerated stand information could be employed in a larger

model or decision support system such as NED-2, providing enhanced what-if analysis upon

simulated disturbance.

3.2   DECLARATIVE KNOWLEDGE BASE

Part of what makes REGEN Agent modular is the fact that declarative knowledge is separate

from procedural knowledge. Procedural (operational) knowledge encompasses the

establishment, sprouting and selection algorithms, the multiple-run handler and statistical metric

calculations, which are collectively termed the core engine, while competitive rankings and

probability coefficients for individuals and plot-wise stand composition are considered

declarative (factual) knowledge and are contained in the management unit knowledge base or are

asserted to the blackboard via the client application (see Figure 3.1).

A management unit knowledge base contains species ranking and probability information

for each species included in the management unit. The schemata for each declarative bit of

knowledge (or fact) are detailed below, where a suffix of A indicates the term is an atom, S

indicates a string, N indicates a number, and L indicates a Prolog list:

```
regen_kb_name(KBID)

regen_kb_id_attrib(KBID,ExpertNameS,EMailS,Add1S,Add2S,Add3S,CommentsS,HighMUS,LowMUS)

regen_species_name(CodeA,NameS,KBID)

regen_species_code(NameS,CodeA,KBID)

regen_ranking(CodeA,SizeA,RankN,KBID)

regen_regenerate_from(CodeA,SourceA,KBID)

regen_stem_parameters(SourceA,CodeA,ParmListL,KBID)

regen_establishment_constant(SourceA,CodeA,SEConstantN,KBID)

regen_establishment_logistic(SourceA,CodeA,SEP1N,SEP2N,SEP3N,SEP4N,SEP5N,KBID)

regen_sprouting_constant(CodeA,SPConstantN,KBID)

regen_sprouting_logistic(CodeA,SPP1N,SPP2N,SPP3N,KBID)
```

Figure 3.2: REGEN Knowledge base facts

The `regen_kb_name` fact encodes the unique knowledge base identification (KBID)

atom, which also happens to be the name of the REGEN knowledge base file (.rkb) representing

the particular management knowledge base. Because the agent can handle multiple management

knowledge bases at the same time, the KBID is essential in differentiating facts from each KBID

because they are loaded into memory together. It is the job of the client (e.g., REGEN for Excel)

15

to provide the agent with the KBIDs representing the various management knowledge bases utilized.

The `regen_kb_id_attribute` fact encodes various properties of the knowledge base used in identification. For example, if a knowledge base is distributed for public use, the author can identify himself or herself with contact information and comments on the knowledge base. The final two arguments correspond to the high- and low-level management unit descriptors, which can be used in selecting the various knowledge bases in both the client and knowledge editor applications.

REGEN Agent utilizes various atomic codes to differentiate among species (in lieu of species name strings). Thus, it is essential to provide facilities for converting between species codes and species names; `regen_species_name` and `regen_species_code` meet this need by taking advantage of Prolog's inherent first-argument indexing to provide quick, bi-directional lookup.

Perhaps the most important knowledge is stored in the `regen_ranking` facts, which serve to store the *<species,size,ranking>* triples mentioned in the abstract REGEN model discussion above. Currently, `SizeA` is an atom from the following set: {g, s, m, l, sp}, corresponding to the five size classes currently handled by the model. `RankN` is a number between 1 and 20, with 1 indicating the highest level of competitiveness.

Special regeneration cases are flagged by the `regen_regenerate_from` fact. `SourceA` is an atom from the set {seedlings, root_suckers}, and the engine first checks this flag before processing the various establishment algorithms. This allows what-if analysis without loss of the actual knowledge encoded in the `regen_stem_parameters` facts in that the client can simply set the `regen_regenerate_from` flag without modifying the

`regen_stem_parameters` data. The `ParmListL` argument is a Prolog list with the following structure: [StemRankingN,NumberStemsToAddN]. Because establishment of seedlings and root-suckers can be stochastic, the next two facts, `regen_establishment_constant` and `regen_establishment_logistic` provide the encoding of the various probability function coefficients as described in Equations 3.1 and 3.2.

The other special regeneration case, sprouting from stumps, is usually determined in the client by virtue of the existence of stump sprout regeneration sources. Thus, a flag is unnecessary. Again, stump sprouting can be stochastic, hence the `regen_sprouting_constant` and `regen_sprouting_logistic` facts.

Because each management knowledge base is essentially an ASCII text file of Prolog facts, they can be asserted directly into a Prolog console. Additionally, they can be directly modified or loaded into a custom knowledge editor such as the REGEN Knowledge Editor.

3.3   CORE ENGINE

Again, declarative and procedural knowledge are separate, and the core engine encompasses those bits of knowledge that perform computation, namely Prolog rules. It may be useful to refer to the next section, Client Interface, for a description of the manipulations to the blackboard performed by the client as well as the final facts provided to the client from the engine. The core engine is divided into three modules: *preserve plots, regenerate stand,* and *consolidate runs.* Figure 3.3 provides a high-level overview of the operation of the engine. First, plot data provided by the client (i.e., plot-wise stand composition) are moved to a protected space in the

17

blackboard by collecting all the individual `plot` facts into a list for use in later restoration.

Then the actual regeneration is performed in the *regenerate stand* module by utilizing the various

`plot` facts, the flow of which is illustrated in Figure 3.3. Finally, after all regeneration runs

have been completed, the results of the regeneration are compiled and analyzed by the

consolidate runs module, which exposes the stand- and plot-level results to the client application.
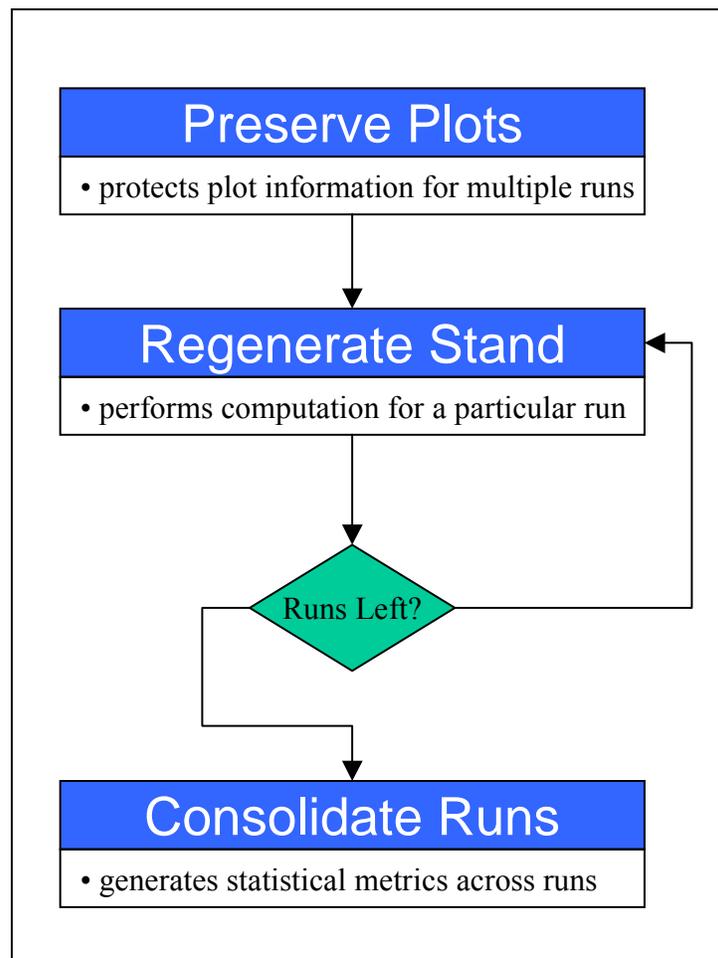

Figure 3.3: Core Engine High-Level Flow

The *regenerate stand* module is composed of several sub-modules as illustrated in Figure 3.4.

Because of the ability to perform the regeneration analysis multiple times, the working space for

the *regenerate stand* module must be reinitialized upon each new run. Reinitialization involves removal of all remaining `plot` facts, renaming of all `regen_regenerated` facts to denote current completed run, and restoration of the original `plot` facts. Essentially, reinitialization removes artifacts and returns the engine to its starting state.

The addition of stump sprouts to the set of `plot` facts is triggered when at least one `regen_potential_sp` fact exists in the knowledge base. This sub-module then checks for the existence of logistic variables provided by the client (within the `regen_potential_sp` fact) and then checks for the existence of the appropriate logistic function and species-dependent coefficients as provided by the appropriate management knowledge base. If both exist and are valid, the number of stump sprouts is logistically determined using the built-in random number generator. However, upon failure of the above, the engine considers constant probability by checking for the species-dependent constant probability coefficient. In the case that neither constant nor logistic parameters are sufficiently defined, the engine skips the stump sprout.

Likewise, the addition of species regenerating from seedlings proceeds in a similar fashion. The engine first checks for the existence of a list of species regenerating from seedlings contained in the fact `regen_special_species_list`. If this fact exists, the engine proceeds to stochastically (as above) add each species to *every* plot in the blackboard by asserting new `plot` facts denoted with the seedling (se) size class and with the number indicated by the `regen_stem_parameters` management knowledge base fact corresponding to species in question.

```
┌─────────────────────────────────────────────┐
│   ┌───────────────────────────────────────┐   │
│   │          Restore Plots               │   │
│   ├───────────────────────────────────────┤   │
│   │ • copies information from protected space │   │
│   └───────────────────────────────────────┘   │
│                    ↓                           │
│   ┌───────────────────────────────────────┐   │
│   │        Add Stump Sprouts             │   │
│   ├───────────────────────────────────────┤   │
│   │ • adds stump sprout ranks based on    │   │
│   │ stochastic information from management │   │
│   │ knowledgebase and client              │   │
│   └───────────────────────────────────────┘   │
│                    ↓                           │
│   ┌───────────────────────────────────────┐   │
│   │          Add Seedlings               │   │
│   ├───────────────────────────────────────┤   │
│   │ • adds species regenerating from seedlings │
│   └───────────────────────────────────────┘   │
│                    ↓                           │
│   ┌───────────────────────────────────────┐   │
│   │        Add Root Suckers              │   │
│   ├───────────────────────────────────────┤   │
│   │ • adds species from root sucker regeneration │
│   └───────────────────────────────────────┘   │
│                    ↓                           │
│   ┌───────────────────────────────────────┐   │
│   │        Regenerate Plots              │   │
│   ├───────────────────────────────────────┤   │
│   │ • applies competition algorithm to each plot │
│   │ yielding plot-level "winning" species │   │
│   └───────────────────────────────────────┘   │
│                    ↓                           │
│   ┌───────────────────────────────────────┐   │
│   │        Expand to Stand               │   │
│   ├───────────────────────────────────────┤   │
│   │ • adds all plot data and converts to  │   │
│   │ trees per acre (TPA)                  │   │
│   └───────────────────────────────────────┘   │
│                    ↓                           │
│   ┌───────────────────────────────────────┐   │
│   │           Clean Up                   │   │
│   ├───────────────────────────────────────┤   │
│   │ • deletes temporary knowledgebase for │   │
│   │ subsequent runs                       │   │
│   └───────────────────────────────────────┘   │
└─────────────────────────────────────────────┘
```
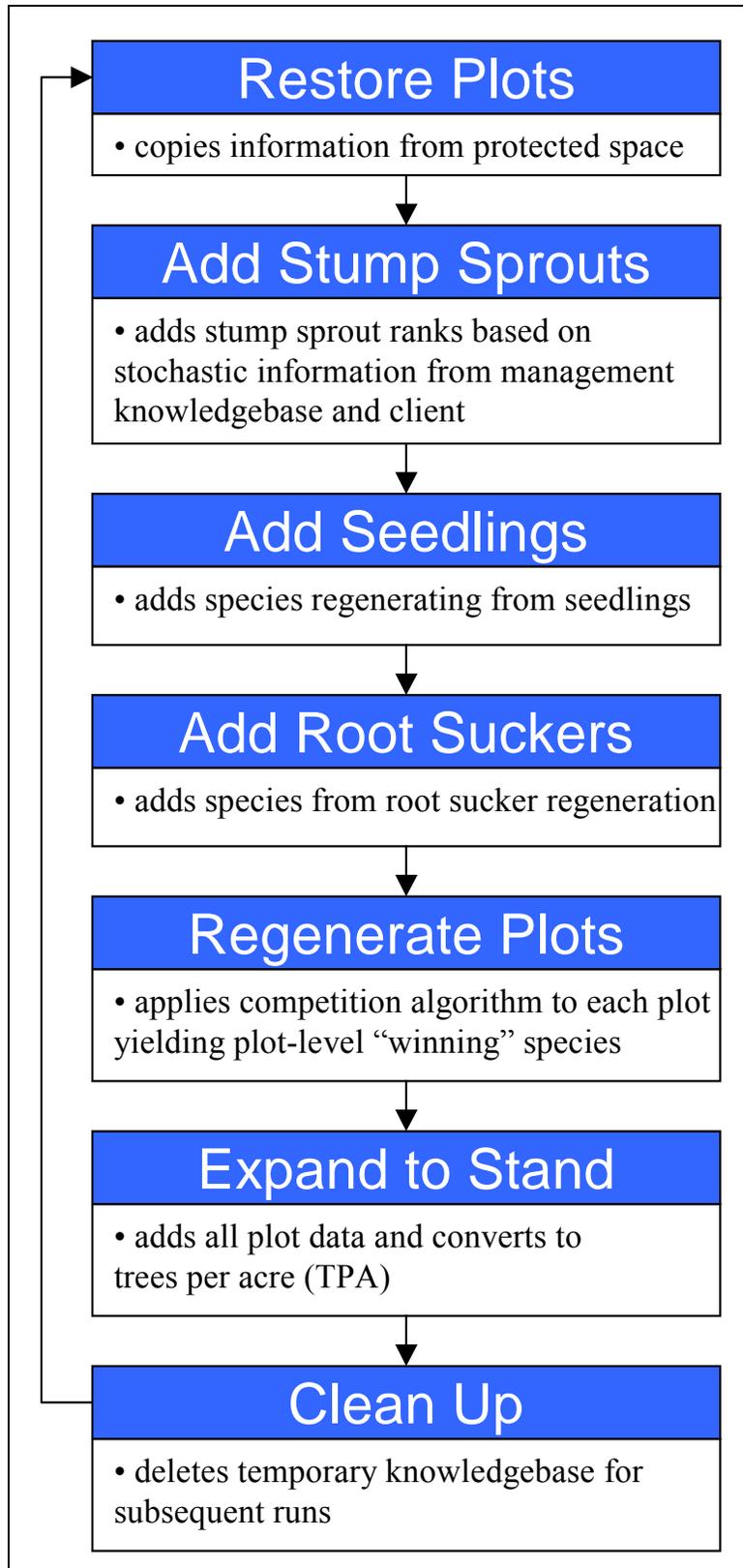
Figure 3.4: Core Engine Regenerate Stand Flow

Addition of species regenerating from root suckers depends on the existence of an advance regeneration individual of the same species in a given plot. Thus, for each species indicated by the `regen_regenerate_from` root-sucker fact, the engine checks for existence of that species in each plot. If the species exists, then root-sucker (rs) individuals are added to each plot according to similar stochastic rules as the regenerate from seedlings case above.

Plot regeneration is the center of the engine. Here, all plot data from both the client application and from the various "species addition" sub-modules are processed according to the Prolog implementation of the Abstract REGEN Model described earlier in this chapter. The engine operates by retracting `plot` facts that match the current top level rank until the selection threshold is met. Upon each successful regeneration event, the given `plot` fact is converted to a `regen_regenerated_plot` fact. When the regeneration process is complete, all regenerated plot facts are consolidated and converted to `regen_regenerated_stand` facts and associated with a run number, and cleanup ensues for the next run. This entire process repeats itself until the desired number of runs is reached.

REGEN Agent provides the client application with several statistical metrics which are generated by the *consolidate runs* module. These currently include variance, standard deviation, sample high and sample low calculations for each regenerated species across all runs. The details of these facts are detailed in the next section.

21

3.4   CLIENT INTERFACE

The client to REGEN Agent is either (1), another modeling or decision support program that incorporates REGEN Agent as part of a larger task or (2), a user interface application providing for stand composition definition and management options. In either case, the interface is identical.

The first task of the client is to provide stand composition information to the REGEN Agent by directly asserting certain facts to the blackboard, as detailed below:

```
plot(PlotIDA,species(CodeA),size(CodeA),count(CountN)
plot_list(PlotListL)
regen_potential_sp(PlotIDA,CountN,DBHListL,SpeciesCodeA)
regen_special_species_list(RegenerateFromSeedlingsListL)
regen_plot_mortality(PMN)
```

Figure 3.5: Input Fact Prototypes

Several `plot` facts are asserted directly by the client and indicate the number of species/size specific advance reproduction individuals that appear on a given plot, while `plot_list` stores the names of each plot (for efficiency purposes). `regen_potential_sp` are client-asserted facts indicating the number and (optionally) sizes (diameter at breast height) of individuals that stochastically sprout from stumps, and `regen_special_species_list` facts are asserted by the client indicating species present on the stand that may randomly disperse seedlings into a plot. The `regen_plot_mortality` fact is a global stochastic variable which indicates a probability that an entire plot could be utterly destroyed and subsequently have no regeneration sources whatsoever.

22

In addition to providing stand composition data to the agent, the client also must handle any stand management options. For example, one management option currently implemented in the REGEN for Excel client application is the removal of regeneration sources according to species/size qualifiers by direct manipulation of the blackboard. Because management options are part of what-if analysis, it was deemed appropriate to include such functionality in the client and not in the agent itself.

Once the agent runs (the client must call the `regenerate_stand_multi(Times)` predicate), several post-regeneration facts are exposed for optional additional processing and/or presentation to the end-user:

```
regen_total_regenerated_tpa(AverageN)
regen_regenerated_stand(species(CodeA),count(AverageN))
regen_regenerated_stand_variance(species(CodeA),VarianceN)
regen_regenerated_stand_standard_deviation(species(CodeA),StdDevN)
regen_regenerated_stand_high(species(Code),HighValN)
regen_regenerated_stand_low(species(Code),LowValN)
regen_survived_plot(PlotIDA,NumberSurvivedN)
regen_regenerated_plot(PlotIDA,species(CodeA),size(SourceA),count(AverageN))
```

Figure 3.6: Output Fact Prototypes

The `regen_total_regenerated_tpa` fact indicates the total number of trees per acre regenerated on the stand, while the `regen_regenerated_stand` facts provide the average stand-level regenerated species count across all runs. For each plot, `regen_survived_plot` provides a count of the number of plots destroyed before regeneration could take place and

`regen_regenerated_plot` yields the average number of species/source individuals on a given plot contributing to the overall stand composition after regeneration.

Because the client interface is relatively straightforward, REGEN Agent becomes a modular and portable software component. While it is not a standalone application, it can be wrapped into a library for use with other programs and programming languages. Logic Programming Associates (LPA) currently provides an extension to their WinProlog development environment called Intelligence Server, which provides for bi-directional calls between a Prolog application and a client application written in any of several programming languages. One of these languages is Microsoft Visual Basic and with some slight modifications, the interface between Microsoft Visual Basic and WinProlog can be used with Microsoft Visual Basic for Applications (VBA), the scripting language for the Microsoft Excel spreadsheet application. It was this capability that drove the development of the REGEN for Excel client application, which is detailed in the next chapter.

Each programming language requires certain syntax. Providing access directly to the agent may be inappropriate at times as unexpected results (and hence errors) may occur. To ease error-handling and interface engineering, a level is desirable between the core REGEN Agent and the client language. The purpose of this code level is to correct syntactic issues, provide for enhanced error handling and debugging, and to augment the collection of Prolog interface functions available to the client language. In the case of the REGEN for Excel client application, the code is written in both the client language (VBA) and in Prolog, given that some procedures are more elegant in a given language. Interface issues between the given client language and Prolog are solved by modifying this level of code; thus, the core engine and knowledge base structure can remain pristine.

CHAPTER 4

REGEN FOR EXCEL

A great many people are familiar with the Microsoft Excel spreadsheet package, which

has seen numerous instantiations in the past two decades. With the advent of Microsoft Office

97, an initiative was made to standardize the macro language between all Microsoft Office

applications in the form of Visual Basic for Applications, or VBA. With newer versions of

Excel, backwards compatibility has been maintained. For more information on engineering

modeling and decision support applications in VBA, please refer to Albright (2001). The

decision to write an interface to REGEN Agent in VBA for Excel was partially motivated by this

unified macro language architecture, although the rapid and visual development capabilities of

VBA played a major role.

REGEN for Excel is a client application utilizing the REGEN Agent regeneration engine

to provide a standalone application for modeling post-disturbance stand composition. It is

written in the Visual Basic for Applications scripting language in tandem with Microsoft Excel

2000, and is compatible with all versions of Microsoft Excel from 97 to 2002. Again, REGEN

for Excel is modular in that it contains three distinct components: the REGEN Knowledge

Editor for creation, modification, and maintenance of REGEN management knowledge bases,

the stand composition editor for input of plot-wise stand composition data, and the report

generator for designation of management options and graphical output of REGEN Agent results.

4.1   KNOWLEDGE EDITOR

The Knowledge Editor is the primary interface by which users can define the species composition and regeneration parameters of a given knowledge base. See Figures 5.1-5.3 in the subsequent chapter for visuals of the various knowledge editor panes. When knowledge entry is complete, the knowledge editor produces a REGEN knowledge base file (a file with the extension "rkb"), which is later utilized by the stand editor for stand composition input and by the core engine. This "rkb" file contains Prolog facts directly corresponding to those defined in Section 3.2. The intent behind the knowledge editor is to provide a convenient, simple method for defining and modifying knowledge bases without recourse to direct text file manipulation, and allows for a relational knowledge structure without requiring the user to memorize species codes. Indeed, there is provision for defining alternative species code sets, although this has not been actualized in the current implementation.

4.2   STAND EDITOR

The Stand Editor is the other declarative knowledge input component that parses information from existing REGEN knowledge bases (i.e., species list and size representation) and provides a method for estimating stand composition by the definition of one or several representative plots. Each plot input sheet (Figure 5.6 in subsequent chapter) is essentially an interface to input plot species/size individual counts and potential variables for logistic equations (i.e., diameter at breast height or "dbh"). Also, a control sheet is provided to access stand editor functions such as creating new plot input sheets, setting global parameters, and invoking the report generator.

26

4.3 REPORT GENERATOR

The report generator is the final component of REGEN for Excel and makes use of several of Excel's features. When the user requests a report generation, the system automatically asserts the information provided from the stand editor session to the blackboard and invokes the core engine. Once the simulation is complete, the report generator queries the blackboard for output facts and formats a report (Figures 5.7 and 5.8 in subsequent chapter). Because Excel has graphing capabilities, various graphs could be implemented by modifying the report generation code. In this implementation, the stand composition pie chart is the default and only chart available. Also, Excel has facilities for HTML output, allowing for web-based display of a given report.

CHAPTER 5

PERFORMANCE AND EVALUATION

5.1   AN EXAMPLE APPLICATION

Perhaps the most appropriate method in which to describe the performance of the REGEN Agent

and REGEN for Excel programs is to step visually through an example application.   Essentially,

there are two classes of end-users: 1), those that wish to utilize an existing management

knowledge base to predict regeneration (a client user, perhaps a landowner or stand manager)

and 2), a forestry expert who wishes to develop a model for a particular area or region by

defining a management knowledge base (an expert user). In any event, the first task is to define

or import a management knowledge base.

A REGEN for Excel session starts by executing the Microsoft Excel program and either

opening an existing REGEN for Excel workbook or creating a new REGEN for Excel workbook

using the template. Figure 5.1 shows the first pane of the knowledge editor, which can be

accessed through the control sheet of the main application (Figure 5.2). If the client user does

not wish to define his or her own management knowledge base, an existing knowledge base can

be imported from a REGEN knowledge base file that has been copied to the "Agent" directory of

the entire system. Once the knowledge base has been imported, it is registered and available to

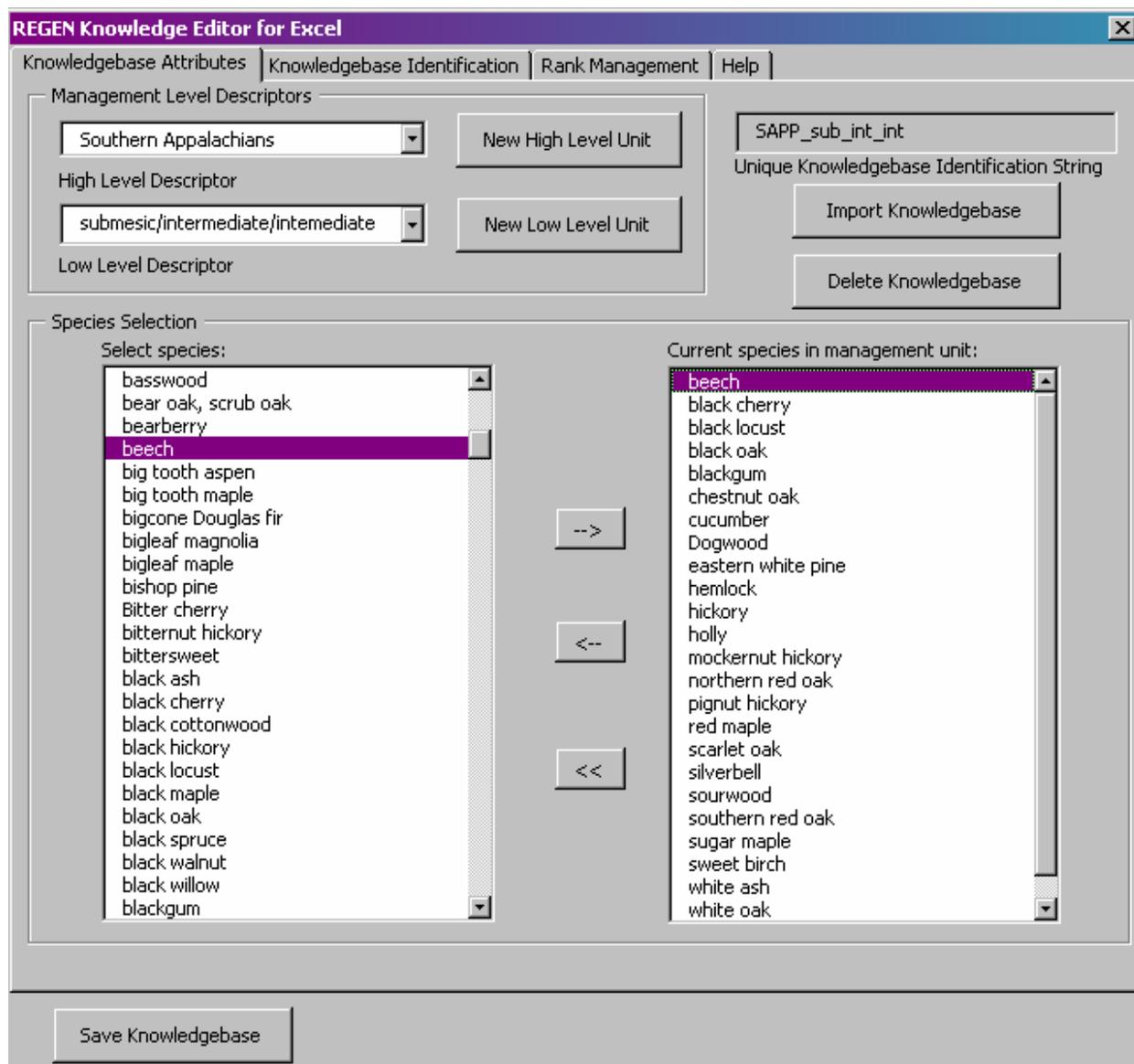the user. A knowledge base can be removed from the set of registered knowledge bases.

Figure 5.1: Editing Knowledge base Attributes

However, this action does not remove the actual knowledge base file from the "Agent" directory, thus providing knowledge base archive and protection against accidental deletion. When the task of defining or importing a management knowledge base is complete, plot-wise stand composition input sheets may be generated and filled with inventory data. When a new plot

Figure 5.2: Control sheet

input sheet is requested, the user is first prompted to select the management unit (i.e., the

management knowledge base) to be associated with this particular plot. This provides additional

flexibility in that stands can be defined which comprise multiple management units with

different REGEN knowledge bases. Theoretically, a user could fill as many plot input sheets as

desired to represent the given stand. Note in Figure 5.3 how species/size combinations that are

undefined in the management knowledge base have their corresponding cells grayed. Also note

that individuals with a DBH of greater than 1.5 feet have an additional field for specifying the

DBH values. This should be a comma-delimited list with a length up to the number of
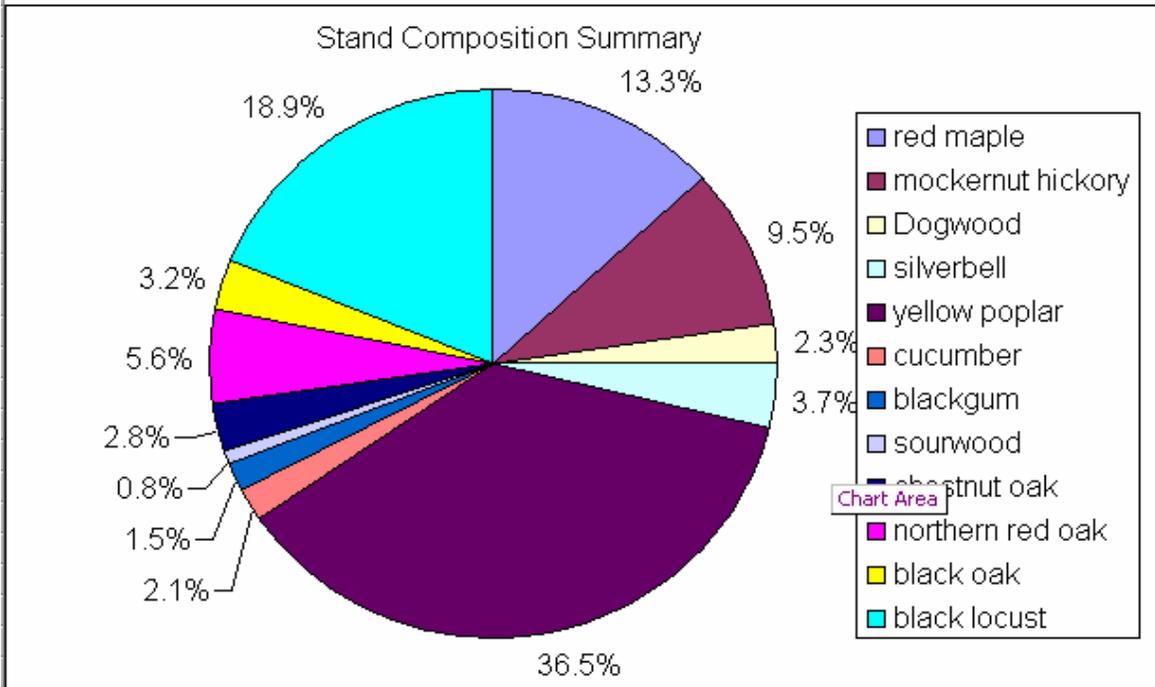
individuals specified in the adjoining field.

Species	Code	Germiant (small seedling)	Small (<= 2 ft)	Medium (> 2 & <= 4 ft)	Large (> 4 ft & <= 1.5 in dbh)	Potential Stump Sprouts (PSS) (> 1.5 in dbh)	DBH Values (must count in PSS)
blackgum	693		3	2		2	2,3
chestnut oak	832		4	1			
mockernut hickory	409		6	2	2		

Stand Name BCTEST1   **Plot Name** 1

Management Units | Southern Appalachians | submesic/intermediate/intemediat | SAPP sub int ii

Figure 5.3: Plot Input sheet

31

Once the stand composition has been declared through filling at least one plot input sheet, the user may generate a report. This action causes all data from the stand composition editor to be asserted into the blackboard of the REGEN Agent, as well as the appropriate REGEN management knowledge bases. The REGEN Agent then executes for the number of runs indicated by Simulation Parameters : Number of Runs setting on the Control sheet (Figure 5.2). Plot mortality can also be defined here, as well as the plot sampling size ($1/100^{th}$ and $1/250^{th}$ acre sizes are common). After regeneration is complete, a report is generated (Figures 5.4 and 5.5) detailing both stand-level and plot-level statistics.

The very first task for the domain expert user is to build a management knowledge base using the REGEN Knowledge Editor. The first step is to start the REGEN Knowledge Editor in the Control sheet (see Figure 5.2). This will bring up the REGEN Knowledge Editor as shown in Figure 5.1. The next step is to define a new management unit by entering in High and Low Level unit descriptors and a unique knowledge base identification code (Figure 5.1). Alternately, the user may elect to modify an existing knowledge base. Next, under Knowledgebase Attributes, the species in the management unit are selected from the master species list on the left. Optionally, the user may define the Knowledgebase Identification fields (Figure 5.7). The final, and most important, knowledge editor task is to define each species' regeneration characteristics such as comparative ranking and sprouting probability under the Rank Management tab (Figure 5.8). Saving the knowledge base gives the stand composition editor access to the new management knowledge base.   The same knowledge base may be edited at any time by simply restarting the REGEN Knowledge Editor. In addition, other knowledge bases may be defined; the user is not limited to one knowledge base for a given stand.

| Regeneration Output for Stand: | BCTEST1 |
| Report ID: | Report 2004-Feb-25 (15.47.53) |

Stand Composition Summary

Legend:
- red maple
- mockernut hickory
- Dogwood
- silverbell
- yellow poplar
- cucumber
- blackgum
- sourwood
- chestnut oak
- northern red oak
- black oak
- black locust

Pie chart values: 13.3%, 9.5%, 2.3%, 3.7%, 36.5%, 2.1%, 1.5%, 0.8%, 2.8%, 5.6%, 3.2%, 18.9%

Simulation Parameters

Number of Runs: 25
Plot-level Mortality: 0.1

Management Options

No management options selected

Summary of Stand

Species	Avg. TPA	Percentage	Std. Error	High	Low
red maple	63.2	13.3%	4.72	100	20
mockernut hickory	45.12	9.5%	6.11	110	26
Dogwood	10.88	2.3%	2.12	48	2
silverbell	17.6	3.7%	0.52	20	20
yellow poplar	173.36	36.5%	14.66	317	34
cucumber	10	2.1%	1.80	20	2
blackgum	7.2	1.5%	10.20	30	30
sourwood	3.68	0.8%	1.96	20	6
chestnut oak	13.12	2.8%	0.78	20	13
northern red oak	26.44	5.6%	1.56	40	20
black oak	15.24	3.2%	4.17	60	7
black locust	89.64	18.9%	8.63	160	15

▶ ▶│ \ Control ⁄5⁄4⁄3⁄2⁄1\ **Report 2004-Feb-25 (15.47.53)** ⁄

Figure 5.4: Generated Report – Stand-level Statistics

Individual Plot Results

5 (20/25) survived

Species	Rank	Avg. # Selected	Source Size
mockernut hickory	6	0.32	medium
yellow poplar	5	2.7	seedling
black oak	5	0.53	medium
black oak	7	0.24	small
black locust	3	1.2	root sucker
black locust	5	0.53	small

4 (24/25) survived

Species	Rank	Avg. # Selected	Source Size
mockernut hickory	6	0.24	medium
Dogwood	5	0.43	large
Dogwood	5	0.11	stump-sprout
yellow poplar	5	1.26	seedling
northern red oak	4	0.96	large
black locust	4	0.96	medium
black locust	3	1.8	root sucker

3 (23/25) survived

Species	Rank	Avg. # Selected	Source Size
red maple	2	1.32	stump-sprout
silverbell	2	0.88	stump-sprout
yellow poplar	5	0.48	seedling
sourwood	4	0.193333333	stump-sprout
northern red oak	4	0.366666667	large

2 (23/25) survived

Species	Rank	Avg. # Selected	Source Size
red maple	7	0.48	small
red maple	2	1.36	stump-sprout
mockernut hickory	8	0.024	small
Dogwood	8	0.016	small
yellow poplar	5	1.44	seedling
cucumber	5	0.52	medium

1 (22/25) survived

Species	Rank	Avg. # Selected	Source Size
mockernut hickory	5	1.333333333	large
mockernut hickory	6	0.36	medium
yellow poplar	5	2.8	seedling
blackgum	6	0.36	medium
chestnut oak	5	0.666666667	medium

▶ ▶|\ Control ∕5 ∕4 ∕3 ∕2 ∕1\ **Report 2004-Feb-25 [15.47.53]** ∕

Figure 5.5 Generated Report – Plot-level Statistics
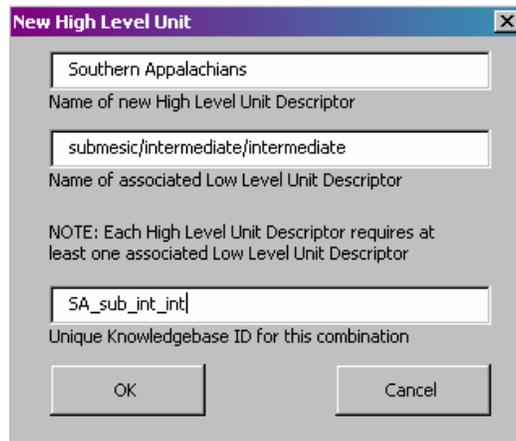
Figure 5.6: Knowledge Editor - New High Level Unit Form
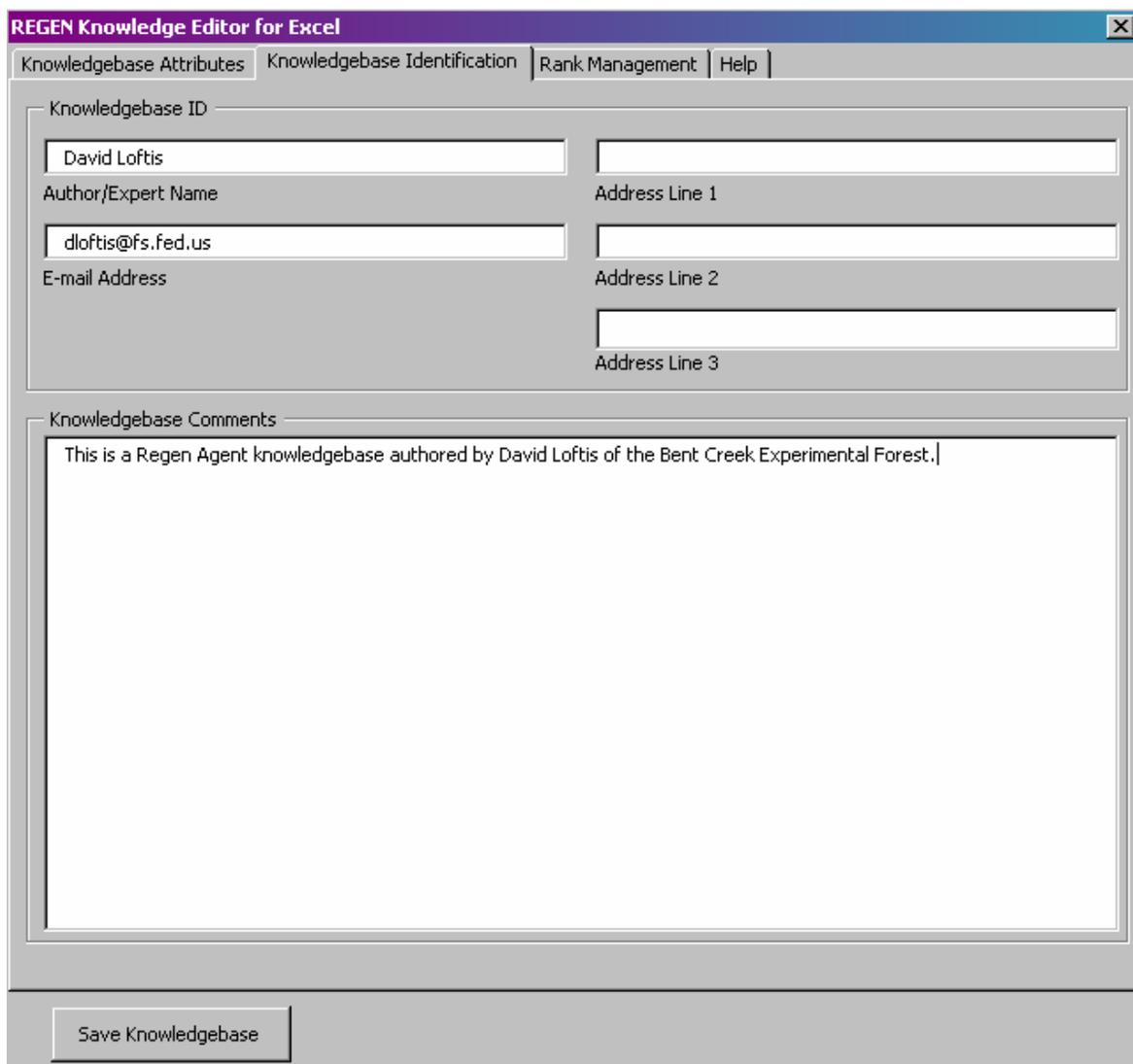
Figure 5.7: Knowledge Editor - Knowledge Base Identification
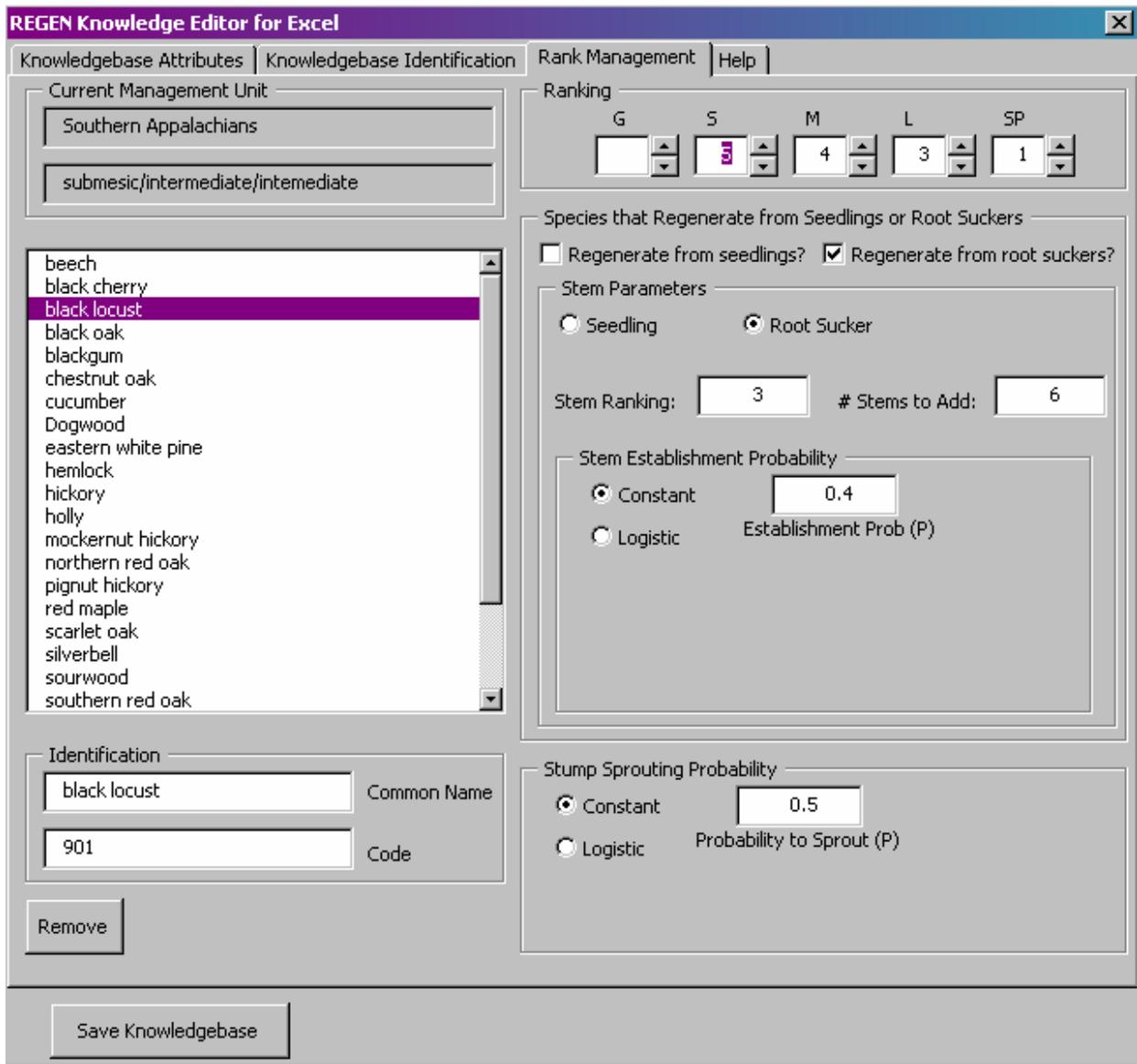
Figure 5.8: Knowledge Editor - Rank Management

5.2   CRITIQUE AND FUTURE WORK

The Loftis model for forest regeneration is becoming popular in forestry circles due to its

generality, the ability to encode expert intuition about regeneration dynamics in a straightforward

way, and the model's utility in analysis of forest management practices (i.e. deliberate

disturbances). This enhancement and implementation improves the utility of the model in that it provides a clear separation of declarative and procedural knowledge, the ability to "swap" and combine declarative knowledge bases between and within simulations, an application interface for "black box" inclusion of the engine into other software, facility to run multiple simulations with many statistical metrics, and an intuitive and familiar user interface utilizing Microsoft Excel. This being said, there is ample opportunity for enhancement of the existing implementation in two domains: the core engine and the Excel user interface. LPA WinProlog is a commercial product and thus any distribution of the core engine is subject to licensing from Logic Programming Associates. However, much development effort has occurred to achieve efficiency and features not found in open source versions. One is the fact that LPA WinProlog is always run as compiled code and is not subject to the performance issues surrounding interpreted programming languages. However, LPA WinProlog still maintains the advantages of Prolog, namely self-modification at runtime, rapid development in that it is a high-level, abstract language, and native functionality in the domains of declarative knowledge management and other specific Prolog features such as unification.

Additionally, the core engine requires storage of temporary information to calculate statistical metrics, in an amount up to 20 kilobytes per individual simulation run. For simulations including many multiple runs (e.g., 1000), this can require a significant engine memory commitment. At this time, standard personal computers should handle this memory requirement easily; however, the underlying LPA Intelligence Server environment does not document a method for modification of the estimated memory requirements when an application is run (i.e., REGEN Agent). Thus, memory must be allocated to the engine at its inception; therefore limiting the number of runs a given simulation can perform. The only recourse to this

problem would be to have the interface start LPA Intelligence Server with an estimate of memory requirements given both the complexity of stand data and the REGEN knowledge bases to be used. Currently, the Excel interface statically allocates 20 megabytes of working memory, allowing sufficient space for a 1000-run simulation at a complexity of about 150 *<species, size>* plot facts. A more complex stand representation (e.g., more plots or more species per plot) would of course proportionally decrease the number of possible runs. Two scenarios could be utilized to decrease the memory requirements of the agent. One is to store information from each run in data files as the data are generated and then reasserting the facts back to the system as they are needed. Another possibility is to store the temporary information in a database such as Microsoft Access, which would be ideal for managing the dynamic amounts of this type of structured data.

The REGEN for Excel interface application is written in Visual Basic for Applications (VBA), which is an interpreted language and subject to the computation drawbacks traditionally associated with interpreted languages. But again, VBA and Excel provide statistical and user interface facilities that would be time-consuming to re-create. During the development of the user interface, it was noted that the messaging system between the LPA Intelligence Server and VBA is slow, yielding significant delays when many rapid queries to the Intelligence Server are required. The solution entails submitting only one query to process and retrieve similar data, such as conversion and retrieval of species names and codes. Another solution would be to include Prolog parsing functionality in the VBA code, allowing VBA to directly access data such as species name / species code entries from the various REGEN knowledgbases and other, static knowledge base files. Yet another solution might involve having the agent export a delimited text file that the client application could then import.

Another drawback to the current implementation of REGEN for Excel is that the VBA code is wrapped into an Excel workbook. Currently, the user creates a new workbook from a template, which copies the VBA code to the newly-created workbook. When deploying a modified or updated version of the REGEN for Excel application, there is currently no method for distributing the code changes to workbooks generated from a previous version's template, nor is there an automated method for exchanging data between versions. Finally, the distribution size of REGEN for Excel is much larger than the previous DSSTools version, which is due almost entirely to overhead introduced by the Excel interface. However, the distribution still fits on a 1.44MB floppy disk and thus is a miniscule download at contemporary Internet connection speeds.

Currently, REGEN Agent is being incorporated into the NED-2 Decision Support System for forest management and the REGEN for Excel application is being evaluated by several interested parties in both the forestry service and the academic community. In particular, an exciting future extension might be a web repository of REGEN knowledge base files so that an end user may download and utilize a particular area's regeneration profile as developed by the regional expert. The goal of this project is that the agent and the standalone application will gain popular acceptance in the forestry community and will both provide practical utility in analyzing regeneration and succession as well as offer pedagogical support in teaching ecosystem and forest management theory.

BIBLIOGRAPHY

Albright, S. Christian. 2000. VBA for Modelers: Developing Decision Support Systems Using

       Microsoft Excel, Duxbury, Thompson Learning.

Deutschman, D. H., S. A. Levin, C. Devine, and L. A. Buttel. 1997. Scaling from trees to forests:

       analysis of a complex simulation model. Science 277:1688.

Dey, Daniel C. 1991. A comprehensive Ozark regenerator. Columbia, MO: University of

       Missouri. 283 p. Ph.D. dissertation.

Dey, D.C.; Johnson, P.S.; Garrett, H.E. 1996a. Modeling the regeneration of oak stands in the

       Missouri Ozark Highlands. Canadian Journal of Forest Research. 26(4): 573-583.

Dey, D.C.; Ter-Makaelian, M.; Johnson, P.S.; Shifley, S.R. 1996b. Users guide to ACORn: a

       comprehensive Ozark regeneration simulator. Gen. Tech. Rep. NC-180. St. Paul, MN:

       U.S. Department of Agriculture, Forest Service, North Central Forest Experiment Station.

       35 p.

Drury, W.H. and Nisbet, I.C.T. 1973. Succession. J. Arnold Arboretum 54: 331-368.

Dixon, Gary E. comp. 2003. Essential FVS: A User's Guide to the Forest Vegetation Simulator.

       Internal Rep. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Forest

       Management Service Center. 193p.

Egler, F.E. 1954. Vegetationj science concepts. 1. Initial floristic composition, a factor in old

       field vegetation development. Vegetatio 4: 412-417.

Ernst, Richard L. 1992. SILVAH: A stand prescription tool. Compiler 10(2):43-45.

Kim, Geneho; Nute, Donald; Rauscher, H. Michael; Loftis, David L. Appbuilder for DSSTools: An Application Development Environment for Developing Decision Support Systems in Prolog. Computers and Electronics in Agriculture. 27 (2000): 107-125.

Kimmins, J.P. 1996. Ecological Succession – process of change in ecosystems. In: Mailly D. ed. Forest Ecology: a foundation for sustainable management. Prentice Hall Amsterdam, Elsevier, 1997.

Loftis, David L. 1988. Regenerating red oak in the Southern Appalacians: Predictive models and practical applications. Ph.D. dissertation, North Carolina State University, Raleigh.

Loftis, David L. 1989. Species composition of regeneration after clearcutting Southern Appalachian hardwoods. In: Miller, James H., comp. Proceedings of the fifth biennial southern silvicultural research conference; 1988 November 1-3; Memphis, TN. Gen. Tech. Rep. SO-74. New Orleans, LA: U.S. Department of Agriculture, Forest Service, Southern Forest Experiment Station: 253-257.

Loftis, David L. 1990. Predicting post-harvest performance of advance red oak reproduction in the Southern Appalachians. Forest Science 36(4):908-916.

Noble, I. R. and Slatyer, R.O. 1980. The use of vital attributes to predict successional changes in plant communities subject to recurrent disturbance. Vegetatio, 43: 5-21.

Nute, Donald; Potter, Walter D.; Maier, Frederick, Wang, Jin; Twery, Mark H.; Rausher, H. Michael; Knopp, Peter; Thomasma, Scott; Dass, Mayukh; Uchiyama, Hajime; Glende, Astrid. NED-2: An Agent-Based Decision Support System for Forest Ecosystem Management. Environmental Modeling and Software. 19 (2004): 831-843.

Oliver, C.D. and Larson, B.C. 1996. Forest Stand Dynamics, John Wiley & Sons, Inc., New York.

Peet, R.K. and Christensen, N.L. 1980. Succession: A population process. Vegetatio 43:131-140.

Peterson, C.J. Computer simulation "BLOWDOWN": Introduction and User's Guide.

    www.plantbio.uga.edu/~chris/simulint.html .

Sander, Ivan L.; Johnson, Paul S.; Rogers, Robert. 1984. Evaluating oak advance reproduction in

    the Missouri Ozarks. Research Paper NC-251. St. Paul, MN: U.S. Dept. of Agriculture,

    Forest Service, North Central Forest Experiment Station.

Shugart, H. H. 1984. A Theory of Forest Dynamics. New York: Springer-Verlag. 278 pp.

Solomon, Dale S. and Leak, William B. 2002. Modeling the regeneration of northern hardwoods

    with FOREGEN. Res. Pap. NE-719. Newtown Square, PA: U.S. Department of

    Agriculture, Forest Service, Northeastern Research Station. 9 p.

APPENDIX A

REGEN FOR EXCEL INSTALLATION AND OPERATING INSTRUCTIONS

REGEN for Excel: System Requirements

REGEN for Excel was developed under Windows 2000 and tested under Windows XP Home and Professional. Certain inconsistencies (i.e., with inter-process communications) between the Windows 9x architecture and the Windows 2000/XP architecture prohibit REGEN for Excel from running on Windows ME, 98 and 95.

Thus, requirements are as follows:

Operating System: Windows 2000 Professional, Windows 2000 Server,
                      Windows XP Home or Windows XP Professional
Excel Version: Microsoft Excel 97 or later versions

REGEN for Excel: Installation

REGEN for Excel is distributed as a self-extracting archive. Installation defaults to the directory "C:\REGEN for Excel" which, at this point, is the required directory. Once the archive has installed, start your version of Excel. Open the file "C:\REGEN for Excel\REGEN for Excel Template.xls". (Depending on your settings, you may see a window pop up requesting you to enable macros. For this software to operate, macros in Excel must be enabled.)

Now, again under the "File" menu, select "Save As". In the "Save as type" drop down box, select "Template (*.xlt)". Ensure that the file name "REGEN for Excel Template.xlt" appears in the "File name" box. Click on "Save". (If the file already exists, simply overwrite with the new template.) Now, close Excel.

REGEN for Excel: Operating Instructions

Given that the operating instructions above have been completed, you are now ready to begin a REGEN for Excel session. Open your version of Microsoft Excel. Under the "File" menu, select "New". In the resulting window, under the "General" tab, you should see a template icon captioned with "REGEN for Excel Template". Select this icon and click "OK", enabling macros if required.

Once the file opens, the worksheet named "Control" will be active. Assuming that at least one REGEN knowledge base is registered, you can begin building plots (i.e., plot input sheets). If

there are no REGEN knowledge bases registered, you will need to start the REGEN Knowledge Editor and create a REGEN knowledge base.

The first task is to assign a name to this stand in the "Stand Name" text box. Then click on the "New Input Sheet" button. Eventually, a window will appear prompting you to select both high and low-level management units for this particular plot. These management or ecological unit descriptors are taken from a registry of unit descriptors maintained by the REGEN Knowledge Editor. Keep in mind that REGEN for Excel allows any number of management units to be assigned to a given stand, with each plot having a designated management unit as defined by the combination of high- and low-level descriptors set at plot sheet creation time.

The first task for the user is to assign a unique name to the plot. Upon completion of management unit selection, the "Plot Name" text box contents are highlighted. Change the name (e.g., "Plot 1") by simply typing in the new name.

Species composition of the plot is defined by selecting cells in the "Species" column. Selecting a cell will cause a list of species descriptors to drop down, from which the user can select any of the species defined in the management unit knowledge base. Upon selecting a species name, the "Code" column is set for that species (e.g., FIA codes, USDA codes, or custom codes as defined by the management unit knowledge base).

In the next plot sheet section (delimited by the vertical double-line), there seven size category columns. For a given species and management unit knowledge base, those size category cells with undefined rankings are shaded. Inputting species counts into these cells will not influence the simulation. In each of the non-shaded cells, the user can input observed individual counts. The final column is associated with the "Count > 1.5 dbh" column. Here, actual dbh values are inputted, comma delimited, corresponding to the count designated in the "Count > 1.5 dbh" column (e.g., "1.6,3.2,4,7"). Repeat this for each species in the plot. The user can add additional plot sheets using the "Control" sheet (or the "Regen" menu on the menu bar) in the same manner.

Once all plots have been designated, the user may then "Generate a Report" using the "Control" sheet or "Regen" menu. If the knowledge base has designations for special species (i.e., species that can contribute seedlings to a plot without being represented in the plot), the user is presented with a window asking for the set of "Special Species" to be included, with parameters designated by the management unit knowledge base(s).

As the report is generated, the user is presented with a summary chart describing projected stand composition. Additionally, summary data and plot-specific data are provided. Because of the stochastic nature of the model, the user may generate several reports using different stand composition and special species settings. Reports are labeled by creation time and date, ensuring no duplication.

Once the session is complete, the user can save the Excel file anywhere on the system (e.g., in "My Documents" or in the "C:\REGEN for Excel" folder).

APPENDIX B

A SAMPLE REGEN KNOWLEDGE BASE

```
% REGEN Knowledgebase File: SAPP_sub_int_int.rkb

regen_kb_id_attrib('SAPP_sub_int_int',`David
      Loftis`,`dloftis@fs.fed.us`,``,``,``,``,`Southern
      Appalachians`,`submesic/intermediate/intemediate`).

regen_kb_name('SAPP_sub_int_int').


% regen_species_name

regen_species_name('129',`eastern white pine`,'SAPP_sub_int_int').
regen_species_name('260',`hemlock`,'SAPP_sub_int_int').
regen_species_name('403',`pignut hickory`,'SAPP_sub_int_int').
regen_species_name('812',`southern red oak`,'SAPP_sub_int_int').
regen_species_name('541',`white ash`,'SAPP_sub_int_int').
regen_species_name('651',`cucumber`,'SAPP_sub_int_int').
regen_species_name('444',`beech`,'SAPP_sub_int_int').
regen_species_name('762',`black cherry`,'SAPP_sub_int_int').
regen_species_name('901',`black locust`,'SAPP_sub_int_int').
regen_species_name('832',`chestnut oak`,'SAPP_sub_int_int').
regen_species_name('833',`northern red oak`,'SAPP_sub_int_int').
regen_species_name('316',`red maple`,'SAPP_sub_int_int').
regen_species_name('806',`scarlet oak`,'SAPP_sub_int_int').
regen_species_name('318',`sugar maple`,'SAPP_sub_int_int').
regen_species_name('372',`sweet birch`,'SAPP_sub_int_int').
regen_species_name('802',`white oak`,'SAPP_sub_int_int').
regen_species_name('693',`blackgum`,'SAPP_sub_int_int').
regen_species_name('590',`holly`,'SAPP_sub_int_int').
regen_species_name('491',`Dogwood`,'SAPP_sub_int_int').
regen_species_name('409',`mockernut hickory`,'SAPP_sub_int_int').
regen_species_name('400',`hickory`,'SAPP_sub_int_int').
regen_species_name('621',`yellow poplar`,'SAPP_sub_int_int').
regen_species_name('837',`black oak`,'SAPP_sub_int_int').
regen_species_name('711',`sourwood`,'SAPP_sub_int_int').
regen_species_name('580',`silverbell`,'SAPP_sub_int_int').


% regen_species_code

regen_species_code(`eastern white pine`,'129','SAPP_sub_int_int').
regen_species_code(`hemlock`,'260','SAPP_sub_int_int').
regen_species_code(`pignut hickory`,'403','SAPP_sub_int_int').
regen_species_code(`southern red oak`,'812','SAPP_sub_int_int').
regen_species_code(`white ash`,'541','SAPP_sub_int_int').
regen_species_code(`cucumber`,'651','SAPP_sub_int_int').
regen_species_code(`beech`,'444','SAPP_sub_int_int').
regen_species_code(`black cherry`,'762','SAPP_sub_int_int').
regen_species_code(`black locust`,'901','SAPP_sub_int_int').
regen_species_code(`chestnut oak`,'832','SAPP_sub_int_int').
```

```
regen_species_code(`northern red oak`,'833','SAPP_sub_int_int').
regen_species_code(`red maple`,'316','SAPP_sub_int_int').
regen_species_code(`scarlet oak`,'806','SAPP_sub_int_int').
regen_species_code(`sugar maple`,'318','SAPP_sub_int_int').
regen_species_code(`sweet birch`,'372','SAPP_sub_int_int').
regen_species_code(`white oak`,'802','SAPP_sub_int_int').
regen_species_code(`blackgum`,'693','SAPP_sub_int_int').
regen_species_code(`holly`,'590','SAPP_sub_int_int').
regen_species_code(`Dogwood`,'491','SAPP_sub_int_int').
regen_species_code(`mockernut hickory`,'409','SAPP_sub_int_int').
regen_species_code(`hickory`,'400','SAPP_sub_int_int').
regen_species_code(`yellow poplar`,'621','SAPP_sub_int_int').
regen_species_code(`black oak`,'837','SAPP_sub_int_int').
regen_species_code(`sourwood`,'711','SAPP_sub_int_int').
regen_species_code(`silverbell`,'580','SAPP_sub_int_int').


% regen_ranking
% species,size,ranking facts

regen_ranking('129',s,6,'SAPP_sub_int_int').
regen_ranking('129',m,4,'SAPP_sub_int_int').
regen_ranking('129',l,2,'SAPP_sub_int_int').
regen_ranking('260',s,8,'SAPP_sub_int_int').
regen_ranking('260',m,6,'SAPP_sub_int_int').
regen_ranking('260',l,4,'SAPP_sub_int_int').
regen_ranking('403',s,8,'SAPP_sub_int_int').
regen_ranking('403',m,6,'SAPP_sub_int_int').
regen_ranking('403',l,5,'SAPP_sub_int_int').
regen_ranking('403',sp,4,'SAPP_sub_int_int').
regen_ranking('812',s,7,'SAPP_sub_int_int').
regen_ranking('812',m,5,'SAPP_sub_int_int').
regen_ranking('812',l,4,'SAPP_sub_int_int').
regen_ranking('812',sp,4,'SAPP_sub_int_int').
regen_ranking('541',s,6,'SAPP_sub_int_int').
regen_ranking('541',m,5,'SAPP_sub_int_int').
regen_ranking('541',l,4,'SAPP_sub_int_int').
regen_ranking('541',sp,2,'SAPP_sub_int_int').
regen_ranking('651',s,7,'SAPP_sub_int_int').
regen_ranking('651',m,5,'SAPP_sub_int_int').
regen_ranking('651',l,4,'SAPP_sub_int_int').
regen_ranking('651',sp,2,'SAPP_sub_int_int').
regen_ranking('444',s,8,'SAPP_sub_int_int').
regen_ranking('444',m,6,'SAPP_sub_int_int').
regen_ranking('444',l,5,'SAPP_sub_int_int').
regen_ranking('444',sp,4,'SAPP_sub_int_int').
regen_ranking('762',g,5,'SAPP_sub_int_int').
regen_ranking('762',s,4,'SAPP_sub_int_int').
regen_ranking('762',m,3,'SAPP_sub_int_int').
regen_ranking('762',l,2,'SAPP_sub_int_int').
regen_ranking('762',sp,1,'SAPP_sub_int_int').
regen_ranking('901',s,5,'SAPP_sub_int_int').
regen_ranking('901',m,4,'SAPP_sub_int_int').
regen_ranking('901',l,3,'SAPP_sub_int_int').
regen_ranking('901',sp,1,'SAPP_sub_int_int').
regen_ranking('832',s,7,'SAPP_sub_int_int').
regen_ranking('832',m,5,'SAPP_sub_int_int').
regen_ranking('832',l,4,'SAPP_sub_int_int').
regen_ranking('832',sp,4,'SAPP_sub_int_int').
regen_ranking('833',s,7,'SAPP_sub_int_int').
regen_ranking('833',m,5,'SAPP_sub_int_int').
regen_ranking('833',l,4,'SAPP_sub_int_int').
regen_ranking('833',sp,4,'SAPP_sub_int_int').
```

```
regen_ranking('316',s,7,'SAPP_sub_int_int').
regen_ranking('316',m,5,'SAPP_sub_int_int').
regen_ranking('316',l,4,'SAPP_sub_int_int').
regen_ranking('316',sp,2,'SAPP_sub_int_int').
regen_ranking('806',s,7,'SAPP_sub_int_int').
regen_ranking('806',m,5,'SAPP_sub_int_int').
regen_ranking('806',l,4,'SAPP_sub_int_int').
regen_ranking('806',sp,4,'SAPP_sub_int_int').
regen_ranking('318',s,7,'SAPP_sub_int_int').
regen_ranking('318',m,5,'SAPP_sub_int_int').
regen_ranking('318',l,4,'SAPP_sub_int_int').
regen_ranking('318',sp,2,'SAPP_sub_int_int').
regen_ranking('372',g,5,'SAPP_sub_int_int').
regen_ranking('372',s,4,'SAPP_sub_int_int').
regen_ranking('372',m,3,'SAPP_sub_int_int').
regen_ranking('372',l,2,'SAPP_sub_int_int').
regen_ranking('372',sp,2,'SAPP_sub_int_int').
regen_ranking('802',s,8,'SAPP_sub_int_int').
regen_ranking('802',m,6,'SAPP_sub_int_int').
regen_ranking('802',l,5,'SAPP_sub_int_int').
regen_ranking('802',sp,5,'SAPP_sub_int_int').
regen_ranking('693',s,8,'SAPP_sub_int_int').
regen_ranking('693',m,6,'SAPP_sub_int_int').
regen_ranking('693',l,5,'SAPP_sub_int_int').
regen_ranking('693',sp,4,'SAPP_sub_int_int').
regen_ranking('590',s,8,'SAPP_sub_int_int').
regen_ranking('590',m,7,'SAPP_sub_int_int').
regen_ranking('590',l,6,'SAPP_sub_int_int').
regen_ranking('590',sp,6,'SAPP_sub_int_int').
regen_ranking('491',s,8,'SAPP_sub_int_int').
regen_ranking('491',m,7,'SAPP_sub_int_int').
regen_ranking('491',l,5,'SAPP_sub_int_int').
regen_ranking('491',sp,5,'SAPP_sub_int_int').
regen_ranking('409',s,8,'SAPP_sub_int_int').
regen_ranking('409',m,6,'SAPP_sub_int_int').
regen_ranking('409',l,5,'SAPP_sub_int_int').
regen_ranking('409',sp,4,'SAPP_sub_int_int').
regen_ranking('400',s,8,'SAPP_sub_int_int').
regen_ranking('400',m,6,'SAPP_sub_int_int').
regen_ranking('400',l,5,'SAPP_sub_int_int').
regen_ranking('400',sp,4,'SAPP_sub_int_int').
regen_ranking('621',g,5,'SAPP_sub_int_int').
regen_ranking('621',s,4,'SAPP_sub_int_int').
regen_ranking('621',m,3,'SAPP_sub_int_int').
regen_ranking('621',l,2,'SAPP_sub_int_int').
regen_ranking('621',sp,1,'SAPP_sub_int_int').
regen_ranking('837',s,7,'SAPP_sub_int_int').
regen_ranking('837',m,5,'SAPP_sub_int_int').
regen_ranking('837',l,4,'SAPP_sub_int_int').
regen_ranking('837',sp,4,'SAPP_sub_int_int').
regen_ranking('711',s,8,'SAPP_sub_int_int').
regen_ranking('711',m,6,'SAPP_sub_int_int').
regen_ranking('711',l,5,'SAPP_sub_int_int').
regen_ranking('711',sp,4,'SAPP_sub_int_int').
regen_ranking('580',s,7,'SAPP_sub_int_int').
regen_ranking('580',m,5,'SAPP_sub_int_int').
regen_ranking('580',l,4,'SAPP_sub_int_int').
regen_ranking('580',sp,2,'SAPP_sub_int_int').
```

```
% regen_regenerate_from
% special species: seedlings and root suckers

regen_regenerate_from('444',root_suckers,'SAPP_sub_int_int').
regen_regenerate_from('762',seedlings,'SAPP_sub_int_int').
regen_regenerate_from('901',root_suckers,'SAPP_sub_int_int').
regen_regenerate_from('372',seedlings,'SAPP_sub_int_int').
regen_regenerate_from('621',seedlings,'SAPP_sub_int_int').




% regen_stem_parameters - seedlings
% species, parameters facts

regen_stem_parameters(seedling,'762',[6,5],'SAPP_sub_int_int').
regen_stem_parameters(seedling,'372',[6,5],'SAPP_sub_int_int').
regen_stem_parameters(seedling,'621',[6,5],'SAPP_sub_int_int').


% regen_stem_parameters - root suckers
% species, parameters facts

regen_stem_parameters(root_sucker,'444',[6,5],'SAPP_sub_int_int').
regen_stem_parameters(root_sucker,'901',[6,3],'SAPP_sub_int_int').


% regen_sprouting_parameters
% both constant and logistic sprouting parameter facts

regen_sprouting_constant('129',0,'SAPP_sub_int_int').
regen_sprouting_constant('260',0,'SAPP_sub_int_int').
regen_sprouting_constant('403',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('812',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('541',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('651',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('444',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('762',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('901',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('832',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('833',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('806',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('318',0.3,'SAPP_sub_int_int').
regen_sprouting_constant('372',0.25,'SAPP_sub_int_int').
regen_sprouting_constant('802',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('693',0.71,'SAPP_sub_int_int').
regen_sprouting_constant('590',1,'SAPP_sub_int_int').
regen_sprouting_constant('491',1,'SAPP_sub_int_int').
regen_sprouting_constant('409',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('400',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('837',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('711',0.5,'SAPP_sub_int_int').
regen_sprouting_constant('580',1,'SAPP_sub_int_int').


regen_sprouting_logistic('832',2.462946,-0.0878581,null,'SAPP_sub_int_int').
regen_sprouting_logistic('833',3.111055,-0.1076234,null,'SAPP_sub_int_int').
regen_sprouting_logistic('316',4.044538,-0.1786859,null,'SAPP_sub_int_int').
regen_sprouting_logistic('806',3.852062,-0.2178389,null,'SAPP_sub_int_int').
regen_sprouting_logistic('318',2.328832,-0.2418001,null,'SAPP_sub_int_int').
regen_sprouting_logistic('372',3.396065,-0.5270189,null,'SAPP_sub_int_int').
regen_sprouting_logistic('802',2.34028,-0.295584,null,'SAPP_sub_int_int').
regen_sprouting_logistic('837',3.153339,-0.2053537,null,'SAPP_sub_int_int').
```

```
% regen_establishment_parameters - seedling
% both constant and logistic seedling establishment parameter facts

regen_establishment_constant(seedling,'762',0.3,'SAPP_sub_int_int').
regen_establishment_constant(seedling,'372',0.3,'SAPP_sub_int_int').
regen_establishment_constant(seedling,'621',0.8,'SAPP_sub_int_int').


% regen_establishment_parameters - root sucker
% both constant and logistic root sucker establishment parameter facts

regen_establishment_constant(root_sucker,'444',0.3,'SAPP_sub_int_int').
regen_establishment_constant(root_sucker,'901',0.3,'SAPP_sub_int_int').
```