Ontological Analysis of Terrain Data

Susanne Riehemann
SRI International
Menlo Park, CA, USA
susanne.riehemann@sri.com

Daniel Elenius
SRI International
Menlo Park, CA, USA
daniel.elenius@sri.com

ABSTRACT

Geographic applications require increasingly accurate data, for example to support high fidelity visual simulations. However, information about data accuracy is typically not directly available, and must instead be inferred from the manner in which the data was acquired and processed. Some inaccuracies arise as subtle side-effects of processing steps, such as transformation errors due to implicit epochs or unintentional downsampling due to pixel overlap of tiled imagery. Many such problems are known to only a small number of experts. To address this problem, we formalize the properties of each piece of data and its processing history in a geographic ontology, and use declarative Semantic Web Rule Language (SWRL) rules to calculate the errors relative to the real world or to other data. Since the impact of these errors depends on the purpose for which the data is to be used, purpose-dependent requirements are described using an additional task ontology and evaluated by our task analyzer software. The geographic ontology combines knowledge from different areas of expertise, and makes it available for the community to use, critique, and augment.

Categories and Subject Descriptors

I.2.4 [**Knowledge Representation Formalisms and Methods**]: OWL and SWRL Ontologies; H.4 [**Information Systems Applications**]: Terrain Data Analysis

General Terms

Theory, Measurement

Keywords

Terrain ontology, terrain accuracy, metadata

1. INTRODUCTION

Military training and testing events make use of many heterogeneous systems and resources, such as simulation programs, virtual trainers, and live training instrumentation. This domain is replete with interoperability problems. One of the most problematic areas is that of terrain representation. For most training and testing purposes, it is crucial for the simulation software to have a realistic representation of the environment in which the events take place, including elevation data, imagery, and 3D models of trees and buildings.

These events are often done in a distributed fashion, where many systems are connected to play different parts in the simulation. The different systems are not usually designed to be used in this fashion, and may use different terrain data and formats. It is difficult to determine whether terrain data is adequate for a particular purpose, because it lacks sufficient metadata, and has often gone through many processing steps, some of which affect accuracy in subtle ways understood only by a handful of experts.

SRI International has been working with military organizations for many years to provide better methods and products to support training and testing events. In the Joint Training Experimentation Program (JTEP), we developed several innovative technologies for so-called Live-Virtual-Constructive (LVC) training for the National Guard [6]. More recently, in the Open Netcentric Interoperability Standards for Training and Testing (ONISTT) and Analyzer for Net-centric Systems Confederations (ANSC) programs, we have developed an ontology-based method for analyzing and planning these events. However, the terrain data problems discussed in this paper are not unique to the military domain, and neither are the solutions we propose.

In Section 2 we describe some of the many problems that can occur with terrain data. In Section 3 we explain why we use ontologies to analyze these problems. Section 4 describes our terrain ontology and rules. In Sections 5 and 6 we discuss lessons learned and possible directions for future work.

2. COMMON TERRAIN DATA ISSUES

One question about terrain data concerns absolute accuracy, i.e. how closely the data matches the real world. This is important for live events where the virtual and real terrains need to match well. If there is no live aspect to the training, one mainly needs to know the relative differences between the virtual terrains for the various roles in the simulation.

The required level of fidelity depends on the use case: a flight simulator does not need as much detail as a terrain that is used for urban training, where even errors in the 1-2 meter range can be significant. Such errors can be caused e.g. simply by ignoring continental drift [9], and can result in line-of-sight (LOS) problems, with buildings being misplaced relative to the live GPS-tracked entities as in Figure 1a.

.

This causes fair fights issues if another virtual terrain in the simulation is correct, as in Figure 1b.

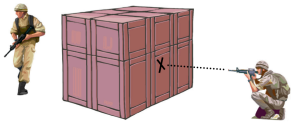

Figure 1a: No LOS with incorrectly positioned building.

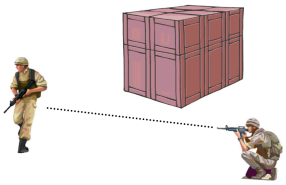

Figure 1b: LOS with correctly positioned entities and building.

Elevation data is often provided as rasters of various resolutions, with each 'pixel' representing one elevation data point. It has frequently been downsampled, reducing accuracy. And not all elevation rasters of a given resolution are created equal. If first return LIDAR data is used without removing the canopy height to produce a bare earth model, or if the raster is created from contour lines, it can contain line-of-sight blocking hills that do not exist in reality. Conversely, significant elevation is sometimes absent due to the large sample spacing. For example, an entire hill is missing in Figure 2a and had to be manually surveyed to create the more accurate model in Figure 2b because the hill was in the middle of a training site. Even if two terrains otherwise use the same source data, they can differ in whether or not such supplementary data is collected, resulting in significant LOS differences between the buildings.

Sometimes elevation data from multiple sources needs to be merged, for example when adding a lower resolution perimeter to a terrain with high resolution elevation data for the training area. Differences in georeferencing can result in artifacts such as the ravine in Figure 3, where the elevation on the left of the ravine is high resolution and ends a little beyond the river, while the elevation on the right is part of the low resolution perimeter.

Many terrain data problems are caused by inadequate metadata. Sometimes there is no metadata at all, and the spatial reference frame (SRF) has to be inferred by reference to other data. The epoch (year) of WGS84 data is almost never stated, making it impossible to correct for continental drift. When State Plane data is provided in feet, it is sometimes unclear whether the units are international feet or US survey feet. Even when the metadata is 'complete' it usually does not maintain a processing history.[1] In particular, the metadata hardly ever contains information about the measurement accuracy of the data acquisition tool or the properties of the data with respect to which georeferencing was performed. An additional complication stems from the fact that different pieces of software store the metadata in different places, e.g. inside a geotiff (a tiff file that includes geographic metadata tags) or in associated sidecar files with various extensions like .aux, .prj, or .tfw. With a processing tool chain involving multiple pieces of software, this can lead to inconsistent information in the different places where the metadata is stored, and can result in errors.

Virtual terrains also use a variety of SRFs. Individual points can be translated between SRFs, so two terrains with different SRFs can be vertex-level correlated. But SRFs differ in the types of distortions they introduce, i.e. distortion of shapes, areas, directions, or distances. For example, augmented UTM (Universal Transverse Mercator) terrains ignore the curvature of the ellipsoid [11], which can lead to LOS issues in large terrains. SRF differences can also interact with dead reckoning algorithms to produce diagonal orientations (crabbing) and zig-zag movement patterns when entities snap to updated positions.

[1]Some software tools like ESRI ArcInfo can record a processing history, but it is not sufficiently complete or human readable, and the information will likely be lost when different tools are used at a later stage in the tool chain.

Figure 2a: Hill missing from 10m elevation data.

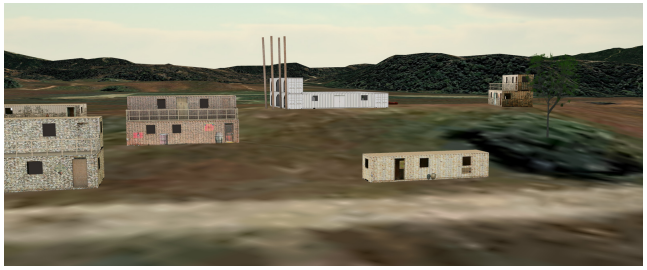

Figure 2b: Hill added using survey data.

Most virtual terrains use a triangular mesh to represent the elevation information, but differ in triangle density, regularity, and distribution of shapes, due to different polygonization algorithms and settings. If there are not enough triangles to model the ruggedness of the terrain, this can lead to LOS issues as well, with entire hills or ravines missing. In one real exercise for which we analyzed the properties of the virtual terrains after the fact, we found a 12 meter worst-case elevation difference between two terrains. These terrains used the same elevation source data – the difference was due to polygon density settings. In one of these terrains, the elevation of the terrain skin was more than 10m above or below the elevation posts of the source data for 15% of the posts, while in the other terrain this was the case for only 0.3% of the elevation posts.

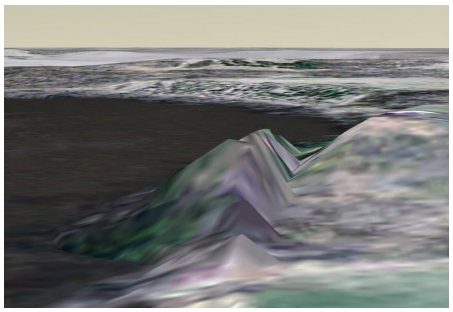

Figure 3: Artificial ravine caused by elevation merging.

3. ONTOLOGIES, OWL, AND SWRL

We use ontologies and the OWL (Web Ontology Language) and SWRL (Semantic Web Rule Language) languages to address the problems described in Section 2. There are several reasons why ontologies, and OWL and SWRL in particular, offer good solutions to many terrain problems.

First, communities can benefit from *open* data representation, as opposed to closed, proprietary data. Open data is reusable in different applications, extensible by third parties, and does not waste investment in the data if one particular application ceases to be used. Closed, proprietary data is usually intended for a specific application and not extensible or reusable. There is a general trend in information technology towards more open standards and data formats, e.g., XML, Web Service APIs, and OWL.

A related distinction is that between *procedural* and *declarative* information. Often, domain knowledge is hidden in procedural code of the form "first do this, then do that". Procedural information is typically closed, as well as difficult to understand and maintain. Declarative information consists of statements of facts, and is typically easier to understand, and more reusable and extensible. Again, the trend is toward using more declarative formats, for example object-oriented programming languages and ontologies.

Third, data needs to be *structured* in order to make it independent of its applications. XML, database schemas, and object-oriented (OO) class models represent the structure of their data. Different data representation methods differ in the degree and type of structure they can capture. For example, XML data is tree-shaped, database content is in table form, and OO models capture class hierarchies. The more sophisticated the application, the more structure we need in the data. Ontologies are highly structured and expressive knowledge representation (KR) schemes. An additional advantage of a highly structured or formalized KR scheme is that it forces domain experts to make hidden assumptions explicit. This makes the knowledge available to others, and can even be beneficial to the experts themselves; we have noted that our own understanding of the terrain domain has been dramatically improved by the systematizing effect of encoding our knowledge in a structured way.

OWL is also *modular*. Ontologies can *import* other ontologies, and refer to concepts defined elsewhere. Each concept has a globally unique resource identifier (URI). This modularity means that the authoring and ownership of ontologies can be highly distributed, with each expert community contributing their knowledge to a global web. Inferences can then be drawn from a combination of knowledge from ontologies produced by different groups of experts, fostering an understanding that transcends individual domains.

There are additional advantages to using formal logic for KR. Formal logic has been used to capture knowledge since Aristotle. Logics come with *inference rules*, allowing us to draw potentially unanticipated conclusions from asserted facts. The asserted facts could all be very simple, yet the inferred facts can be complex and non-obvious. Because the inference rules are strict if-then rules, their application can be automated. Automated inference is also known as *machine reasoning*, and automated inference systems are usually called *reasoning engines* or *inference engines*.

There are many different logics, e.g., Horn logic, first-order logic, modal logics, higher-order logic, which differ in their expressiveness. More expressive logics come at a price: the automated inference requires more time and computing resources. Description Logics (DL) are a family of logics that represent a good trade-off between expressiveness and tractability for many types of applications.

OWL is a standardized DL language that has gained wide acceptance in different communities. There are many freely available tools and reasoning engines for OWL. The basic building blocks of OWL are classes, properties, individuals, and axioms that describe how these relate to each other. SWRL is an extension that makes it possible to write rules expressing mathematical and other relationships that cannot be described in plain OWL.

Ontologies have been used for other purposes in the geo community: to facilitate the interoperability of different GIS systems [12, 5], to help reason about spatial relationships [14, 13], to directly contain geospatial data on the Semantic Web, and to facilitate spatial web searching [7]. There are papers about the properties that these ontologies should have, and how to ensure their interoperability [10, 4]. In contrast, our ontology formalizes metadata, including the processing histories of elevation, imagery and culture data and of the virtual terrains produced from this data, and captures mathematically how each type of processing step affects accuracy, depending on the properties of the input and output data. Our knowledge bases do not contain geographic data directly but instead contain detailed metadata, which we use to compute the absolute errors of the geographic data compared with the real world and relative to other data.

4. A TERRAIN ONTOLOGY

We present our approach to some of the problems discussed in Section 2, using the technologies described in Section 3. Our solution is based on the following ideas: 1) each piece of terrain data, whether it is a fully functional virtual terrain or some component data (such as the positions of buildings in the terrain), is described in an ontology; 2) this ontology includes information about the *processing history* of the data; 3) this information allows us to calculate the maximum or typical *errors* of the terrain data, relative to the real-world terrain or to some other terrain data; and 4) if we know the amount of error in terrain data, we can determine whether the data is suitable for a particular task. In the following, we elaborate on these ideas and show how we implement them. It should be noted that, while our examples are from the military domain, the problems and solutions are also applicable to other domains.

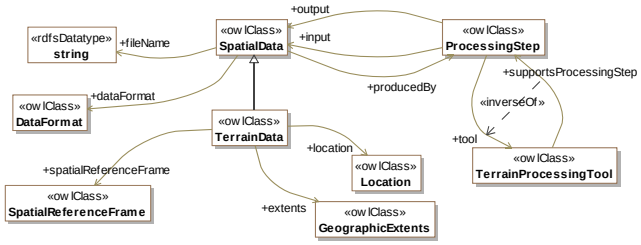

Figure 4: Top level of the terrain ontology.

Most of our terrain ontology resides in one OWL file, `virtual_terrain.owl`. The top-level classes and the relationships between them are shown in Figure 4. The `SpatialData` class represents virtual terrains and any kind of data that can be used to create a virtual terrain, such as elevation data, satellite and aerial imagery, and survey points. It includes

spatial data that is not yet georeferenced, such as 3D models. `SpatialData` has a `DataFormat`. `TerrainData` represents spatial data that is geo-located. Thus, this class has among its properties a spatial reference frame and geographic extents. The `ProcessingStep` class represents one discrete step that takes `SpatialData` as input and produces different `SpatialData` as output (e.g., downsampling imagery creates new imagery with a lower resolution). Thus, the class has `input` and `output` properties. `SpatialData` has a `producedBy` property which is the inverse of the `output` property: `SpatialData` that is `producedBy` a given `ProcessingStep` is the `output` of that `ProcessingStep`. A `ProcessingStep` uses some `TerrainProcessingTool`. Our `terrain_tools.owl` ontology contains information about some common terrain tools, with classes like `ESRI_ArcInfo` and `Presagis_TerraVista`.

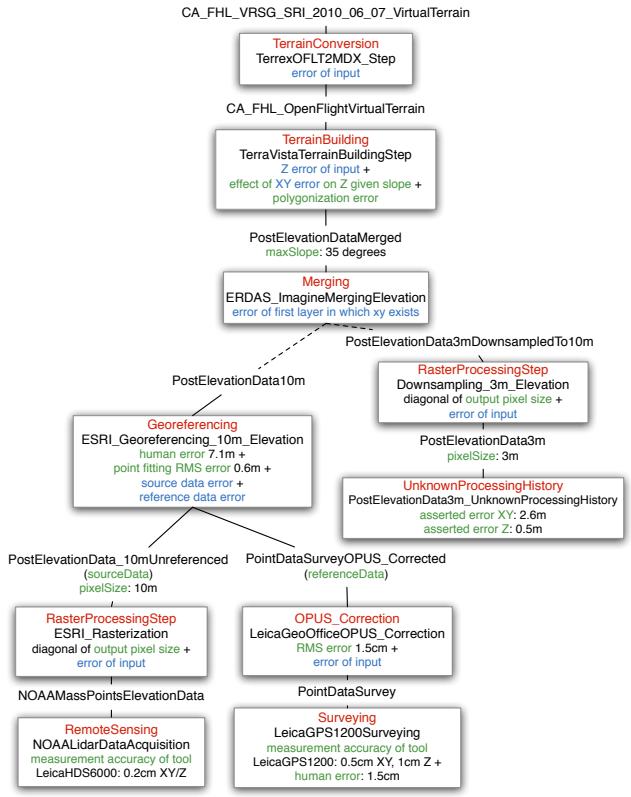

Figure 5: Processing history for Ft. Hunter Liggett elevation data.

The most detailed part of our ontology describes the processing steps. The `ProcessingStep` class has a hierarchy of sub-classes for different kinds of processing steps. For each kind of processing step, there are several SWRL rules that describe the effect of the processing step on the error of its output. Typically, this depends on the quality of the input data and of some properties of the processing step itself. We will discuss these rules in Section 4.1. Figure 5 shows a diagrammatic representation of a typical processing history for the elevation data of a virtual terrain. The specific information about this terrain is stored separately from the general virtual terrain ontology, in an OWL file that imports `virtual_terrain.owl`. The boxes show individuals of processing step classes, including the class name (e.g., `RasterProcessingStep`), the individual name (e.g., `ESRI_Rasterization`), and the formula for calculating the error of the output (e.g., diagonal of output pixel size + error of input). Each box is connected to its input and output terrain data individuals (e.g., `NOAAMassPointsElevationData` and `PostElevationData_10mUnreferenced`). Some of the terrain data individuals have additional properties (e.g., `pixelSize` 10m).

4.1 Terrain Processing Rules

The SWRL rules for calculating the error of terrain data form a complex web of dependencies (see Figure 6). Each rule is a *Horn clause*, i.e., a logical formula of the form $\forall \bar{\varphi_h}, \bar{\varphi_1} \ldots \bar{\varphi_n} : B_1(\bar{\varphi_1}) \wedge \ldots \wedge B_n(\bar{\varphi_n}) \implies H(\bar{\varphi_h})$. This should be read as an if-then rule in which $B_1(\bar{\varphi_1}) \wedge \ldots \wedge B_n(\bar{\varphi_n})$ is the *body* or *antecedent* and $H(\bar{\varphi_h})$ is the *head* or *consequent* of the rule. If the body is true, then the head has to be true. Note that the body has several subformulas (and several predicates B_i) combined using conjunction (logical "and"), whereas the head has only one formula (and one predicate H). The $\bar{\varphi_i}$ are sequences of *variables*. The variables are universally quantified on the outside of the whole rule, but in the SWRL syntax this is assumed by default and therefore omitted. Usually we read the rules backwards, as "in order to prove $H(\bar{\varphi_h})$, we have to prove $B_1(\bar{\varphi_1}) \wedge \ldots \wedge B_n(\bar{\varphi_n})$". The rules essentially *define* the predicates in the rule heads. Note that many rules can have the same predicate in their rule heads. Figure 6 shows which rules define which predicate in our terrain ontology. For example, we can see that the `maxErrorMeters` predicate (bottom middle) is defined by a large number of rules, such as `MaxErrorGeoreferencingZ` and `MaxErrorRasterXY`. The different rules for the same rule head predicate can be thought of as different *cases*. The `MaxErrorGeoreferencingZ` rule defines how `maxErrorMeters` is calculated for the case of a `Georeferencing` processing step and for the Z error. The `MaxErrorRasterXY` rule defines how the error is calculated for the case of a `Rasterization` processing step and for the X/Y error. Figure 6 also shows dependencies between rules, i.e. when a rule has a predicate in its body which occurs in the head of another rule. Sometimes a rule depends on its own head predicate, i.e. the rule is *recursive*.

In Figure 5, we have English descriptions of what some of the rules do. We mentioned the rasterization step, which calculates the error of the output terrain data as "diagonal of output pixel size + error of input". This calculation is defined in the `MaxErrorRasterXY` SWRL rule in the following way (slightly simplified from our actual rule):

$$
\begin{aligned}
&\texttt{producedBy}(?td, ?ps) \wedge \\
&\texttt{RasterProcessingStep}(?ps) \wedge \\
&\texttt{pixelSize}(?td, ?pi) \wedge \\
&\texttt{swrlm:sqrt}(?sr, 2) \wedge \\
&\texttt{swrlb:multiply}(?ep, ?pi, ?sr) \wedge \\
&\texttt{input}(?ps, ?in) \wedge \\
&\texttt{maxErrorMeters}(?ei, Horizontal, ?in, ?x, ?y, ?time) \wedge \\
&\texttt{swrlb:add}(?e, ?ep, ?ei) \\
&\Rightarrow \\
&\texttt{maxErrorMeters}(?e, Horizontal, ?td, ?x, ?y, ?time)
\end{aligned}
$$

We read this backwards, starting with the rule head (the last line after the implication arrow). The `maxErrorMeters` predicate has six arguments (see Section 5 for a discussion of predicate arity): $?e, Horizontal, ?td, ?x, ?y, and\ ?time$. Note that SWRL requires all variables to be prefixed with a question mark. $?e$ is the result variable, the error of the terrain data under the circumstances given by the other parameters. $Horizontal$ is an individual (a constant) indicating that this is the rule for the horizontal error. A similar

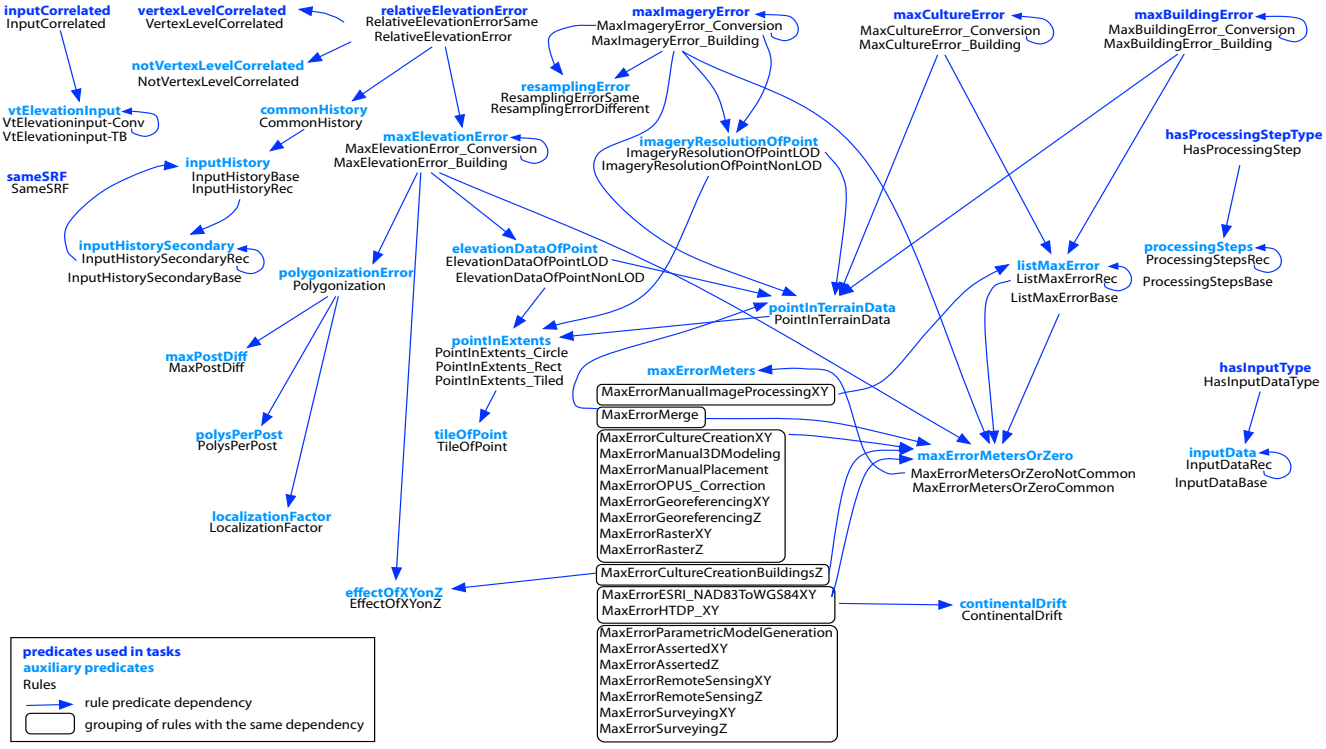

Figure 6: SWRL rules and their interdependencies.

rule covers the vertical error case. ?td is the terrain data that we wish to know the error of. ?x and ?y represent the coordinate of the specific point in the terrain at which we want to know the error. This matters because terrains can have insets with different resolutions. For example, Figure 7 shows a non-rectangular gaming area in which the middle blocks have three levels of detail (LODs) with 1 m resolution imagery and more triangles than the dark 2 m resolution two LOD blocks or the outside 4 m resolution blocks with only one LOD. ?$time$ is the time of the event during which the terrain will be used. This matters when we consider errors caused by continental drift over the years.

Next we look at how the rule body calculates the error. The first line, `producedBy`(?td, ?ps), retrieves the processing step ?ps that produced the terrain data ?td. The next line, `RasterProcessingStep`(?ps), checks that ?ps is the type of processing step that this rule handles, `RasterProcessingStep`. Next we get the `pixelSize` ?pi of the terrain data. Then we calculate the square root of 2 and assign the result to the variable ?sr. Next we multiply the pixel size by the square root of two and store the result in the variable ?ep (think of this as e_p, the "pixel error"). Next we get the input ?in of the processing step. Then we get the error of that input data, using the same ?x, ?y, and ?$time$ values as in the head of the rule, and store the result in the ?ei "input error" variable. Finally, we add the pixel error to the input error and assign the result to ?e, the total error.

Note the recursive use of `maxErrorMeters`. This rule will invoke another rule to calculate the error of the input, and that rule will in turn depend on other rules, until we reach a final piece of data that does not depend on any previous steps. This will be reflected by a non-recursive rule for that processing step. Note also that `swrlm:sqrt`, `swrlb:multiply`, and `swrlb:add` are so-called SWRL built-ins (see Section 5).

Not all processing steps introduce additional error. For example, the HTDP (Horizontal Time Dependent Positioning) processing step reduces the error by the amount of continental drift it corrects for. It is also a good example of a processing step (from the geodetic community) that is not generally known in the GIS (Geographic Information Systems) or military training communities.

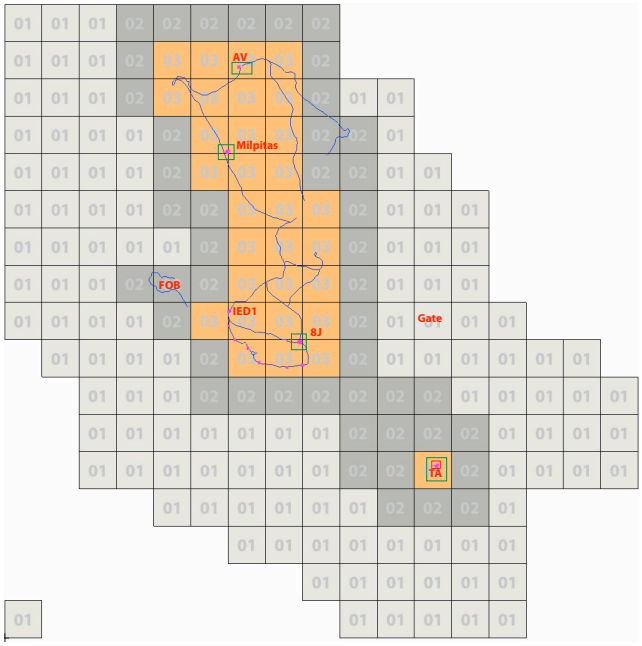

Figure 7: Gaming area with inset LODs.

In some cases a processing step is widely known, but some of its properties are not. For example, few users of ESRI's Arc software know that the output of the transformation *NAD_1983_To_WGS_1984_5* has an epoch of 1996, and even fewer Terra Vista users know that four pixels on each side of an imagery texture tile are 'overlap' and should not be included when calculating the output resolution. This is also an example of natural language being so ambiguous that it is important to state the fact formally. Someone else might well describe it as an 'eight pixel overlap on each side', because for each edge, eight pixels are present in both of the adjacent textures. It is much clearer to formally state how many pixels per texture (eight) to subtract from the total pixel dimensions to get the number of actually used pixels.

For individual terrains our rules currently compute the vertical accuracy of the elevation, the horizontal accuracy of the imagery, and both horizontal and vertical accuracy of the culture data. Thanks to modern survey tools and OPUS (Online Positioning User Service) corrections for atmospheric distortions, the corners of buildings, doors, and windows can theoretically have 2 cm accuracy. Wrong culture survey points can affect LOS out of windows and between buildings, so the computation of these errors is particularly important, and relatively small errors like a few years' worth of continental drift can be significant.

Our rules also detect potentially problematic elevation merging or manual processing steps, and data with unknown processing history or missing WGS84 epoch information. The exact amount of error introduced by these steps cannot be quantified automatically, but our system can output a warning and explain what needs to be verified manually.

When comparing the relative errors between two terrains, we also keep track of the history of all the data used to create the terrains, and discount the errors in any piece of data that is common between the terrains, i.e. we ignore the errors introduced by the data or processing steps that are the same. The rules can also detect whether two terrains both use orthometric/geoid or ellipsoidal heights (known as the 'big H/little h problem'), and whether they are input-correlated (created from the same input data) and/or vertex-level-correlated (created by the same terrain building step). Note that even vertex-level-correlated terrains can differ in points between the vertices due to mismatches in SRFs.

These computed properties of the terrains are compared with the requirements of the particular tasks for which the terrains are intended, using our task analyzer for purpose-aware reasoning [2]. For example, the terrain accuracy requirements of a flight simulator training task might be lower than those for an urban ground training task, and a task that needs two terrains may have specific correlation requirements depending on the planned types of interactions between the simulations. Each specific requirement, e.g. horizontal building corner accuracy, tree height accuracy, or correlation level, is formalized as a constraint on the task, including a severity for violating the constraint, so that a combined quality score relative to the task can be computed. This is important when more than one resource is available to potentially play a role in a task: our analyzer software ranks the possible assignments of resources to roles by their overall scores. The terrains are not the only relevant resources for these tasks: there might also be constraints on hardware, communication infrastructure, etc.

5. DISCUSSION

5.1 The Knowledge Acquisition Bottleneck

One well-known problem in the Knowledge Representation field is the so-called *knowledge acquisition bottleneck*. To be effective, our approach requires a large amount of domain knowledge to be formalized into ontologies. The problem has several components: a) domain experts may provide incomplete, inconsistent, or incorrect knowledge; b) domain experts may not be able to articulate their knowledge; and c) domain experts may not understand the ontology language, and/or ontology experts may not understand what the domain expert is trying to explain.

We can distinguish two types of knowledge acquisition: *building ontologies* and *populating knowledge bases (KBs)*. The building of ontologies consists of creating a structure of classes and properties, axioms (restrictions) that relate these to one another, and rules. Population of KBs consists of creating individuals that instantiate the classes in the ontology, and specifying how the individuals are connected to each other. Building an ontology requires both a deep understanding of the domain and the ability to abstract from specific details to situate specialized knowledge from a sub-field in a larger framework. Even though we were lucky to have significant domain expertise in-house, our terrain ontology building effort required our domain expert to learn about ontologies, and our ontology expert to learn about the domain, which took a significant amount of time. This will not always be practical as we branch out our ontology to other areas.

Populating the KBs is more straightforward than building the ontology. Here, the bottleneck is the volume of information that needs to be entered into the KBs. The distinction between ontology building and KB population is somewhat blurry – often one discovers flaws in the ontology while populating the initial KB, creating a back-and-forth workflow between the two tasks. Subsequent KBs of a similar type should require fewer or no revisions to the ontology.

Strategies for alleviating the knowledge acquisition bottleneck naturally depend on good tools and methodologies. Unfortunately, we have found existing tools lacking in stability, user-friendliness, and features. We posit that ontology building will always require an ontology expert who understands the details of the formal language being used. It is not unreasonable to expect ontology experts to be familiar with tools that are less than ideal. Therefore, we feel that the ontology building task suffers more from lack of good methodologies than lack of tools. Populating the KBs, on the other hand, should be possible for domain experts on their own, given an appropriate user interface. Here, we feel that the lack of good tools is the main problem. Protégé [8] for both knowledge acquisition tasks. Recently, however, we have developed and started to experiment with our own simplified tool that non-ontologists can use to create KBs. In many cases, it is also possible to populate KBs automatically, using domain-specific tools that import data from other sources. We have recently created such "importers" for several types of non-terrain data. This type of automation is only possible if useful metadata is present in the source format. In the case of terrain data, the metadata needs to include the relevant information about the processing history. We will continue our work to produce improved tools, as our efforts move from in-house proof-of-concept towards a transition to real use in the field.

5.2 Limitations on Language Expressiveness

For the main ontology, OWL is sufficiently expressive. In fact, we make relatively light use of most of its constructs. We do, however, make heavy use of SWRL rules, especially in our terrain ontology. SWRL allows us to do mathematical computations, and define other complex relationships that are not part of OWL itself. However, we have run into several limitations of SWRL, which we discuss briefly in the following. They are explored in more detail in [3].

SWRL requires user-defined predicates to have only one or two arguments because regular OWL classes (unary) and properties (binary) are used as predicates. This effectively limits the language to functions of one variable, because one argument position has to be used for the result of executing the function. There are two ways around this problem, neither of them fully satisfactory. The first is to use an RDF (Resource Description Framework) list to contain several arguments. This makes the rules very verbose and mired down in representation details that make them hard to understand and change. It also means that one cannot "pattern match" on the rule head, which could otherwise make rules more elegant. The second solution is based on the fact that SWRL has so-called built-in predicates which can take an arbitrary number of arguments. In Protégé, we can create our own pseudo-built-ins simply by creating new individuals of the `swrl:Builtin` class. These can be used as n-ary predicates in rules, and fully defined in SWRL itself with no need for an external "built-in" definition. This is the solution that we have adopted, but it is less than ideal because it is unlikely to work in tools and inference engines other than our own.

A second serious limitation of SWRL is the lack of *closed-world reasoning*. OWL and SWRL adopt an *open world assumption*, which means that it is always possible that a fact may be true, even if it is not currently known to be true. The assumption is that we do not know everything there is to know, which is quite reasonable for Semantic Web applications with distributed sources of knowledge. However, it causes difficulties for writing rules, because it is too hard to prove a negation under the open-world assumption – we cannot prove that a fact is not true as long as it is at all possible that the fact is true. This is called "classical negation". Some languages, like Prolog, have a "closed-world assumption," and along with it a different form of negation called "negation as failure" (NAF), where a fact is considered to be false if it cannot be *proven* to be true. In other words, one only reasons with locally known information, which makes negations easy to prove. While we cannot entirely drop the open-world assumption, having some form of *local closed-world assumption* is crucial, in our experience. For now we have adopted the following solution: we have added a new built-in predicate `allKnown` which returns a list of all the *known* values for a given individual and property. A list is inherently a closed-off collection on which we can conveniently perform various kinds of computations. Again, this is not an ideal solution – reasoning engines have to implement this new built-in for our rules to work (of course, our own engine is currently the only one that does implement it), and a more general form of local closed-world reasoning would be preferable.

A third limitation of SWRL is its inability to *create new individuals* as a result of evaluating rules. SWRL can create new relationships between individuals, or calculate numbers, but cannot create new individuals. For example, we use individuals to represent *quantities* – entities that have a value and a unit (e.g., "5 meters" or "2 lbs"). We can use SWRL to determine if one quantity is greater than another, but we cannot *add* two quantities together – this would require creating a new individual. This limitation is perhaps the most fundamental of the issues discussed here. For the moment we use quantities as inputs to our rules, but produce plain numbers with assumed default units as outputs.

In spite of these limitations, using OWL and SWRL is currently the best option. We argued in Section 3 that it is important to use open standards for this type of problem.

6. FUTURE WORK

Some aspects of the terrain ontology itself would benefit from further work and input from other subject matter experts (SMEs), e.g. in polygonization, georeferencing, orthorectification, projections and transformations, coordinate representation and translation. We do not claim to have a definitive formalization of each of these processing steps – instead we provide a framework in which they can all be included. It would be particularly valuable if people who are familiar with the algorithms that are used by common tools could formalize the effects of those algorithms at the necessary level of detail. Ideally each geo-processing tool would come with specifications of how it affects the horizontal and vertical accuracy of the output depending on relevant settings and properties of the input data. An independent ontology based system like ours could then import these specifications, evaluate the tools' claims, add rules for steps that do not include them, combine this information, and compare it against the requirements of a given task.

For each candidate formalization it will also be necessary to verify its accuracy empirically. In some cases, such as terrain simplification, there is previous work on the subject [1], and experimental results can be obtained relatively easily because they only concern the relative accuracy between electronic data. In other cases, where accuracy with respect to the real world is concerned, high precision survey data, high resolution LIDAR scans or imagery data may need to be collected for comparison.

While our knowledge bases do not directly contain geographic data, but rather geographic *metadata*, we do need to use geographic coordinates to describe and reason about the extents of the data and the locations for training events. This introduces some difficulties due to the fact that extents and locations may be described in different SRFs. For the moment we represent coordinates as a pair of numbers assumed to be meters in the relevant UTM zone. It would be more general and accurate to use latitude/longitude representations, but those are hard to compute with, converting to/from the UTM coordinates of the actual data is complex, and SWRL is not well suited to the task. A celestiocentric representation is a good compromise in that it can represent points anywhere, but is still in metric space. The existing conversion tools do not currently handle epoch-specific WGS84 data, and it would also be harder to distinguish between horizontal and vertical errors, but it is probably the best choice for our ontology.

One could automatically detect suspicious sequences of processing steps like first downsampling and then upsampling the same piece of raster data, which can sometimes happen unintentionally, hidden in the texture and block size settings of the terrain building step and the quirks of the ter-

rain building software. For example, Terra Vista does not use the full area of a texture but generates some undocumented overlap, as described in Section 4.1. Similarly, bugs in specific versions of specific pieces of software could be formally documented by one SME, to automatically alert others to potential problems. For example, we currently have a problem with OpenFlight terrains being assigned a different origin than the (correct) origin of their (0,0) tile, which took a while to diagnose and might have gone undetected.

In addition to computing the (mostly additive) worst case error, one could compute the more typical RMS (root mean square) error for cases where the individual errors are independent of each other, and keep track of the covariances for correlated data. If a terrain does not meet the requirements of a given task, the system could suggest possible alternative processing steps that would reduce the error. One could also formalize the properties, settings, and algorithms of the software that uses these terrains for display or calculations. Entities may be ground-clamped, their movements may include dead reckoning, LOD switching distances may be affected by LOD range scale settings, clipping planes may be changed, satellite imagery textures may be downsampled or compressed, etc.

More generally, a detailed and precise ontology is an ideal place to formally specify the geospatial terminology of different communities. For example, one group's 'coordinate system' is another's 'spatial reference frame', of which the 'coordinate system' is just one component. One community's 'geodetic' is another's 'geographic'. 'WGS84' usually refers to a datum but sometimes also a geographic coordinate system, a geocentric coordinate system, an ellipsoid, or the system including all of the above and more. The SEDRIS project already formalized most of these concepts, and we have created a SEDRIS ontology, but have not yet specified the correspondences between SEDRIS concepts like 'object reference model' and e.g. ESRI terminology.

7. CONCLUSIONS

We constructed a formal ontology of the properties and processing histories of terrain data, including rules quantifying the errors introduced by each type of data processing step. The ontology and rules bring together detailed domain knowledge from several different areas of expertise and make this expert knowledge available for the computation of relative and absolute errors. Our task analyzer software compares the errors to the requirements for a given task, in order to adjudicate the suitability of the geographic data.

One important aspect of having a formal ontology (for any domain) is that the knowledge is made explicit and can be used, critiqued, and improved by the community. As the complexity and accuracy demands of geographic applications increase, this type of in-depth analysis of terrain data becomes essential. We must make sure that adequate metadata is available for this purpose.

Acknowledgments

The work described in this paper was carried out at the SRI facilities in Menlo Park, CA and was funded in part by the U.S. Department of Defense, TRMC (Test and Evaluation/Science and Technology) T&E/S&T (Test and Evaluation/Science and Technology) Program under NST Test Technology Area prime contract N68936-07-C-0013. The authors are grateful for this support and would like to thank Gil Torres, NAVAIR, for his leadership of the Netcentric System Test (NST) technology area, to which the ANSC project belongs. We would also like to acknowledge ODUSD/R/RTPP (Training Transformation) for its sponsorship of the associated ONISTT project, and thank the three anonymous COM.Geo reviewers for their detailed and helpful comments.

8. REFERENCES

[1] D. Andrews. Simplifying terrain models and measuring terrain model accuracy. Technical Report TR-96-05, University of British Columbia, 1996.

[2] D. Elenius, R. Ford, G. Denker, D. Martin, and M. Johnson. Purpose-Aware Reasoning about Interoperability of Heterogeneous Training Systems. In *The Semantic Web, ISWC 2007 + ASWC 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 750–763. Springer, 2007.

[3] D. Elenius, D. Martin, R. Ford, and G. Denker. Reasoning about Resources and Hierarchical Tasks Using OWL and SWRL. In *International Semantic Web Conference*, volume 5823 of *Lecture Notes in Computer Science*, pages 795–810. Springer, 2009.

[4] F. Fonseca, G. Câmara, and A. M. Monteiro. A Framework for Measuring the Interoperability of Geo-Ontologies. *Spatial Cognition & Computation*, 6(4):309–331, 2006.

[5] F. T. Fonseca, M. J. Egenhofer, P. Agouris, and G. Câmara. Using Ontologies for Integrated Geographic Information Systems. *Transactions in GIS*, 6(3):231–257, 2002.

[6] R. Ford, V. Lamar, R. Giuli, and S. Oberg. The Joint Training Experimentation Program Approach to Distributed After Action Review. In *EURO Simulation Interoperability Workshop, 04E-SIW-063*, 2004.

[7] C. B. Jones, R. Purves, A. Ruas, M. Sanderson, M. Sester, M. van Kreveld, and R. Weibel. Spatial information retrieval and geographical ontologies. In *SIGIR '02*, pages 387–388. ACM, 2002.

[8] H. Knublauch, M. A. Musen, and A. L. Rector. Editing Description Logic Ontologies with the Protégé OWL Plugin. In V. Haarslev and R. Möller, editors, *Description Logics*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.

[9] S. Z. Riehemann. Coordinate Systems and Terrain Reusability. In *Simulation Interoperability Workshop, 09F-SIW-089*, 2009.

[10] P. D. Smart, A. I. Abdelmoty, B. A. El-Geresy, and C. B. Jones. A Framework for combining Rules and Geo-ontologies. In *1st International Conference on Web Reasoning and Rule Systems*, 2007.

[11] R. M. Toms and P. A. Birkel. Choosing a Coordinate Framework for Simulations. In *Simulation Interoperability Workshop, 99F-SIW-183*, 1999.

[12] U. Visser, H. Stuckenschmidt, G. Schuster, and T. Vögele. Ontologies for geographic information processing. *Computers & Geosciences*, 2002.

[13] M. Wessel. Some Practical Issues in Building a Hybrid Deductive Geographic Information System with a DL Component. In *KRDB*, volume 79, 2003.

[14] R. Winkels, R. Hoekstra, and E. Hupkes. Normative reasoning with geo information. In *COM.Geo*, 2010.