Cross-drain Placement to Reduce Sediment Delivery from Forest Roads to Streams

Florentiu Damian

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2003

Program Authorized to Offer Degree: College of Forest Resources

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Florentiu Damian

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____
Peter Schiess

_____
Bruce Lippke

_____
Gerard Schreuder

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purposes or by any means shall not be allowed without my written permission.

Signature_____

Date_____

University of Washington

Abstract

Cross-drain Placement to Reduce Sediment Delivery from Forest Roads to Streams

Florentiu Damian

Chair of the Supervisory Committee:
Professor Peter Schiess
Management and Engineering Division, College of Forest Resources

Ditch relief culverts can reduce road sediment delivery to streams by allowing infiltration and sediment filtering across the forest floor. Below the last ditch relief culvert, all the sediment routed by the ditch will be delivered directly to the stream. The last ditch relief culvert should be as close to the stream crossing as possible. If the ditch relief culvert is too close to the stream however, then there is little potential for sediment filtering. This tradeoff between minimizing the amount of water delivered directly to the stream and maximizing the distance for outflow filtration poses a question of where the last ditch relief culvert should be placed.

A model has been developed which allows a designer to place ditch relief culverts at various locations and subsequently evaluate their impact on sediment delivery to streams. The main feature of the model is its immediate feedback to the forest engineer in visual as well as quantitative form. It allows the designer to dynamically assess the sediment impacts associated with each culvert as it is placed on the road network. Sediment delivery and routing algorithms are based on accepted methodologies. Current as well as planned roads can be evaluated and the potential for improvements documented in a quantifiable and repeatable way.

The model was tested on a portion of the Tahoma State Forest, situated south of Mt. Rainier. Two existing road systems with 28 and 39 stream crossings and 82 and 86 cross

drain culverts respectively, were analyzed. Interactively relocating 20 and 35 of the cross drains resulted in a three quarter reduction in sediment delivered to the stream system. The last culvert was usually placed about 100 - 200 ft of a stream according to local conditions, challenging one of the regulatory recommendations to place a cross drain within 100 ft of a stream crossing. Forest engineers and regulators now have a design tool to assess effectiveness of a cross drain system rather then simply relying on culvert spacing and count.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

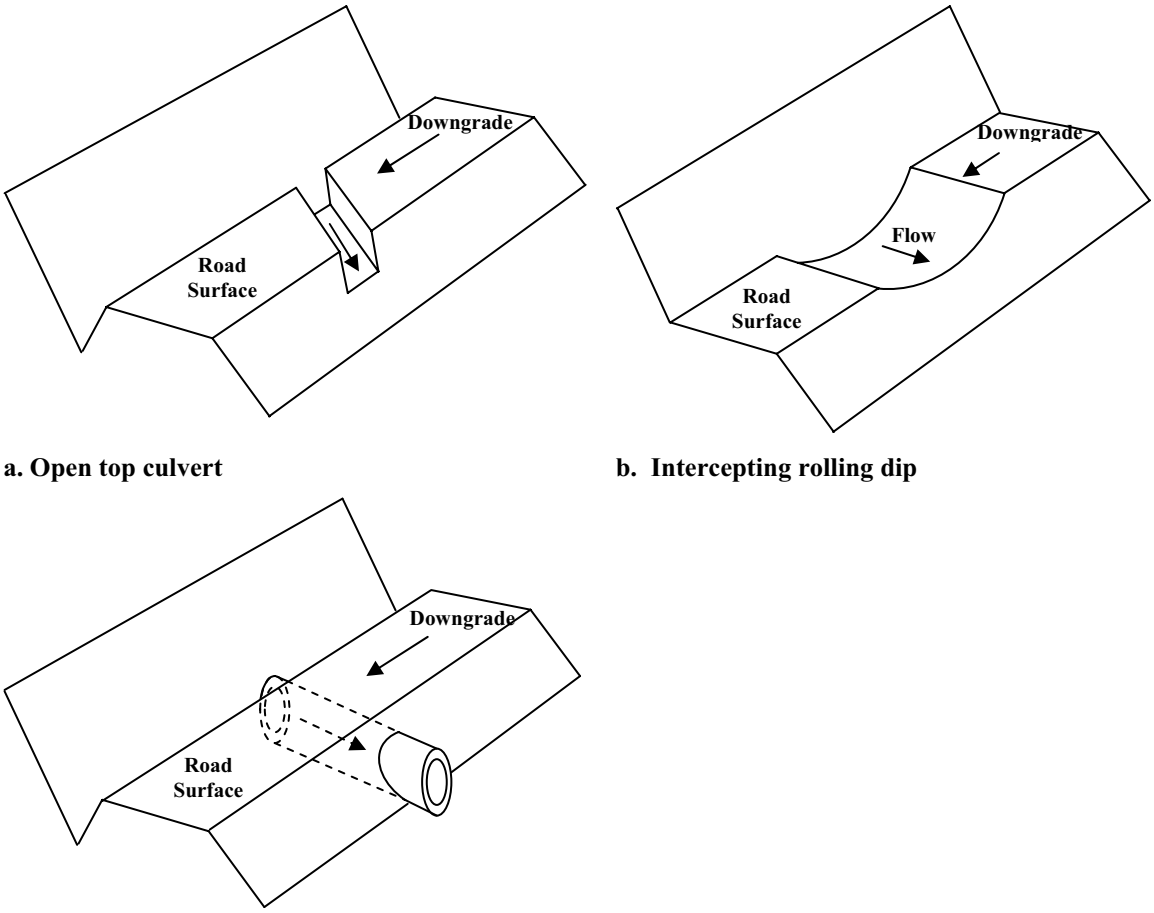
The Road Sedimentation Problem

Road Prism Structure and Drainage Patterns

Forest roads are at the core of modern forestry, providing quick, economical access to the wooded areas. To facilitate forest operations in areas managed for timber production, a well developed network of roads must exist. Timber harvesting and forest management activities drive the expansion of the existing networks into the inaccessible parts of watersheds and sub-basins. As engineered structures on the landscape, forest roads have the potential to disrupt the drainage characteristics of the watersheds they traverse by altering their natural flow patterns. Water from precipitation fallen on the exposed road surface, moving under the influence of gravity, follows a new path dictated by the local road grade. The cuts required by the basic structure of a road prism across a hillside also capture overland flow in close vicinity of the road tread. The accumulation and movement of this water within the roadway is detrimental to the well functioning of the road (*Schiess and Whitaker 1986*). Problems ranging from erosion of the traveling surface to saturation of the sub-grade and mass failure stem from these circumstances. Such processes are usually accelerated during storm events as larger volumes of water move at higher velocities, developing more destructive energy. In order to keep the road prism in good condition and avoid structural damage, roads are outfitted with drainage features.

Three essential prism components determine the water flow within the roadway: the road crown, the road ditch and the cross-drains. The crown represents the side sloping of the road surface. Its main role is to disperse water laterally away from the tread and prevent harmful flow routing along the travel surface. The side ditch, when present, collects and routes water longitudinally along the road alignment towards the nearest stream crossing or cross-drain structure. The cross drains are routing elements that empty the side ditch and redirect its accumulated water across and away from the road prism, onto the side slope, where it will reenter the natural flow regime. Frequently, cross drains are

implemented by drainage pipes built into the road bed but other, less common implementations exist as well (Figure 1).



a. Open top culvert

b. Intercepting rolling dip

c. Pipe culvert

Figure 1: Schematics of common cross drain types

According to their drainage configuration forest roads can be classified in the following categories: insloped with a ditch, crowned with a ditch, outsloped with ditch and outsloped with no ditch (Figure 2). Each of these types generates different drainage patterns and impacts the original watershed drainage accordingly.

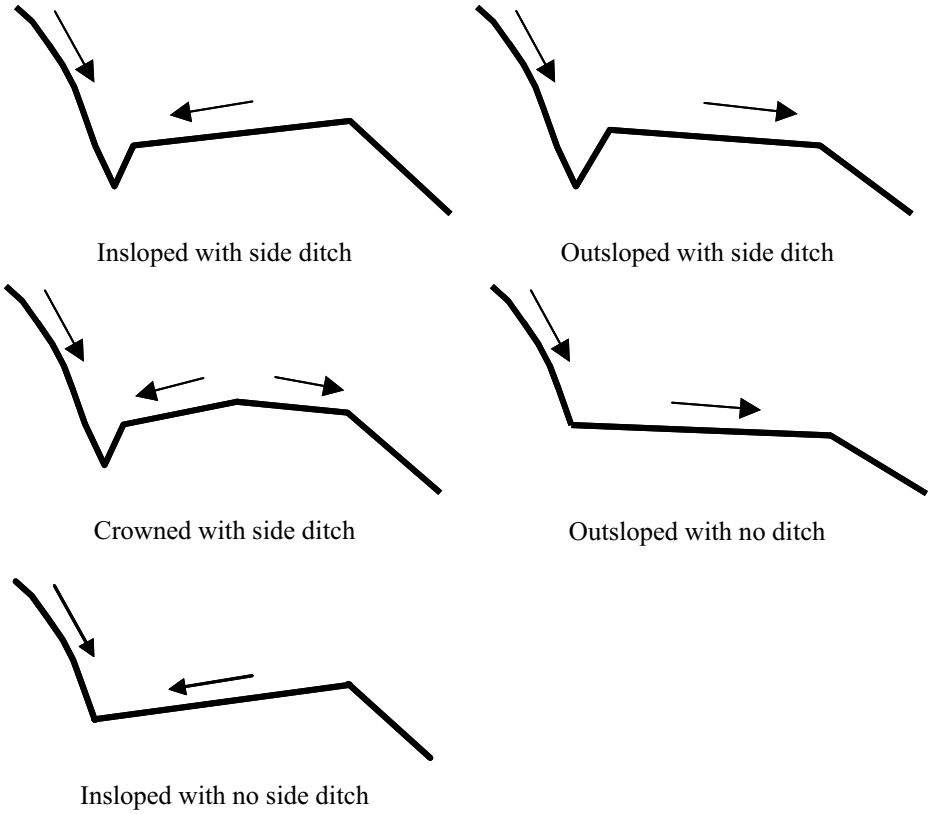


Figure 2: Cross sectional prism types with their drainage patterns

The insloped roads with a side ditch are a common occurrence among forest roads due in part to their ease of maintenance and increased traffic security. The drainage system of an insloped road involves a side ditch, cross drains and stream crossings. The surface of the road, being sloped inwards toward the cut slope, reroutes all water it captures to the side ditch. Consequently, the surface runoff intercepted by the cut bank is concentrated with the rerouted road capture. This can potentially lead to the accumulation of high volumes of water in the ditch. If no cross drains are present this water spills out at the end of the ditch, often directly into a stream at the nearest stream crossing. In cases where the road grade is relatively high, the energy of the moving water can reach potentially destructive levels. Cross drains are placed along the road alignment at various locations to empty the side ditch and reduce the possibility of water induced road and environmental problems.

At the opposite pole of road prism drainage types are the outsloped roads with no side ditch. The outlsoped roads with no ditch disperse all the water they capture on their fill slope. In the absence of a side ditch, concentration and rerouting of large volumes of water does not take place. As the energy of the flowing water is kept to lower levels, its damaging potential is also reduced. Moreover, the lack of direct ditch drainage into a stream network induces less hydrologic disturbance. Harmful processes of erosion and sedimentation generated by moving water are minimal. From an environmental standpoint, the outsloped road type represents a more suitable design. However, traffic security concerns raised by log trucks and heavy equipment, critically restrict its usability. Outsloping is normally applied to minor roads and spurs.

Other intermediary road types inheriting physical characteristics from both the insloped and outlsoped roads do exist (Figure 2). Their hydrologic impacts on the environment are closer to one or the other base types described above, with variations dictated by their elemental differences.

Sediment Production and Delivery Mechanisms

Stream sedimentation is an environmental problem generated by the expansion of sediment into streams in excess of the natural amount from hill-slope erosion and soil creep. Forest roads have been identified as a major contributor to sedimentation. Fine sediment generated by roads is transported into adjacent streams, leading to degradation of water quality and damage to aquatic habitat (*WA Forest Practices Board 2000*). Sedimentation from roads can be approached from two major perspectives: sediment production and sediment delivery.

Sediment production refers to the physical generation of sediment, the process of detaching soil and parent material particles under the influence of various agents. Different premises govern the generation of sediment on each individual road component. The cut and fill slopes are subject to surface erosion occurring when detachable soils are

exposed to erosion factors such as: overland flow, raindrop splash, freeze-thaw, dry ravel, and other biogenic processes (*WA Forest Practices Board 1997*). The litter cover, typically present in a forested environment, protects the soil against all these factors by dissipating erosive energy before it reaches the surface. Road construction operations however, expose the soils on the cut and fill banks, increasing the potential for particle detachment. New roads tend to generate significant quantities of sediment for the first few years of their existence but in time, as side slopes re-vegetate, the sediment production drops. The road's running surface is another major sediment producer. Traffic is the destructive agent acting upon the surface material. The action of tires against the road surface will grind and dislocate particles into smaller units that can eventually be carried away by other agents like water and wind. Some researches consider traffic to be the single most essential factor influencing sediment generation, capable of increasing sediment amounts by one order of magnitude or more (*Reid and Dunne 1984*). The side ditch can also be considered as a source of sediment where the erosion is caused by moving water. In the cases where water gains sufficient energy to overcome the soil's sheer strength, particle detachment is observed.

Sediment delivery refers here to the transportation of sediment generated by the road prism into neighboring stream networks. Fine sediment travels in suspension, carried by overland flowing water along its paths towards the river system. As an active component of watershed hydrology, the road drainage has a fundamental impact on the sediment delivery process. By determining the local water flow within and about the roadway, the road drainage implicitly controls the sediment routing (*Section: Road Crowning and Drainage Patterns*). The various road drainage configurations in conjunction with the actual road location drastically affect the amount of sediment reaching the stream. For example, in the cases of roads that have numerous water crossings, the presence of a side ditch establishes a direct connectivity between the road drainage and the stream network. The side ditch typically empties at stream crossings, unloading all accumulated sediment into the water. This configuration results in a high potential for stream sedimentation.

Other high delivery scenarios include valley bottom roads that run parallel and within close distance to a stream. Large quantities of sediment can reach the stream laterally through surface runoff and water diverted from existing cross drain culverts. On contrast, roads with an outsloped surface geometry and ridge top roads located far away from any streams, have minimal impacts on sedimentation and can be considered as disconnected from the natural watershed drainage.

An important aspect when examining overland delivery on lateral slopes is the site's filtering potential. Through filtering, parts of the sediment produced by the road are deposited before they have a chance to enter the stream network. Filtering is a complex process, fundamentally based on energy loss and infiltration. The carrying water's transport capacity is proportional to its velocity. By reducing water velocity soil particles are dropped from suspension and settle down. The porosity of the soil surface influences the infiltration rates and further contributes to the deposition and absorption of fine sediment. In practice, a reduction of sediment delivery is obtained by diverting the water onto vegetated side slopes where the litter layer, plants, and gentle slopes can slow down the surface runoff enough to trigger these filtering effects. Various researchers have shown that the nature of the parent material and the local micro-topographical features have a major influence on the distance the sediment can travel downhill on a side slope. In some extreme cases these distances can be fairly significant (*Ketcheson and Megahan 1996*), but usually the filtering effects are seen within the first 200 ft from the road (*WA Forest Practices Board 1997*).

Current Management Techniques for Reducing Sediment Impacts from Forest Roads

Contemporary forest practices regulation requires protection of the neighboring streams from harmful road generated sediment. Although no quantification of minimum acceptable impacts is provided, it is desired that water quality and aquatic habitat not be affected. Ideally no sediment from roads would be delivered to streams (*WA Forest*

Practices Board 2000). In order to meet regulation requirements several road management techniques are effectively applied.

A large part of these techniques address the sedimentation at the production end, acting directly onto the sediment producing factors:

- Road resurfacing is the process of replacing a highly erodible road surface like native dirt or thin gravel with a more traffic resistant surface such as thick gravel or pavement. This method can be very effective but also labor intensive and expensive.
- Road gating reduces traffic during the non-logging seasons. Unused roads are less likely to produce sediment; however the major part of the sediment is still generated during the high traffic season.
- Road abandonment and decommissioning are methods that alter the road prism in order to restore the natural drainage patterns. With abandonment a road is prepared for an extended period of stagnation by removing stream crossings culverts, closing it to traffic and outfitting it with water bars. Decommissioning implies the destruction of the entire road prism, returning the side slope to an approximate natural state. A drastic decrease in sedimentation can be obtained this way.
- Re-vegetation represents an accelerated stabilization of the road's cut and fill banks as means to reducing sediment production especially during the first years of the roads existence when these parts are more active.

Other methods are centered on the actual sediment transport, designed to decrease sediment impacts by obstructing the physical flow of sediment into streams:

- Sediment trapping is a popular technique of filtering ditch water prior to spilling into the streams at stream crossings. The sediment traps are manmade devices that intercept sediment, typically through decantation, and require periodical maintenance to assure optimal functionality. This method has been proved

effective especially when combined with other sediment reducing means described above.

Cross Drain Systems and Sediment Reduction

Cross drain systems have originally been devised as an engineered solution for reducing the adverse effects of excess water within the roadway. They are commonly composed of culverts placed at key locations along the road alignment that drain the side ditch of its accumulated water. Positioning and spacing of these culverts are essential for keeping the road prism in a well functioning state. With the recent evolution of road design into a more environmentally aware paradigm, a new challenge for the cross drain systems has surfaced. Due to their intrinsic properties of intercepting and rerouting sediment-laden ditch water, cross drain systems emerged as a potential solution to the stream sedimentation problem. Thus, in addition to the functionality dictated by the prism health state, another prerequisite was added: to reduce sediment delivery to stream networks.

Designing cross drain systems to meet this new functionality asks for an analysis of the potential amount of sediment delivered by each culvert. In order to reduce sediment delivery, a cross drain must divert the sediment-laden water from the road ditch onto the side slope where it can be dispersed and filtered prior to reaching a stream (Figure 3).

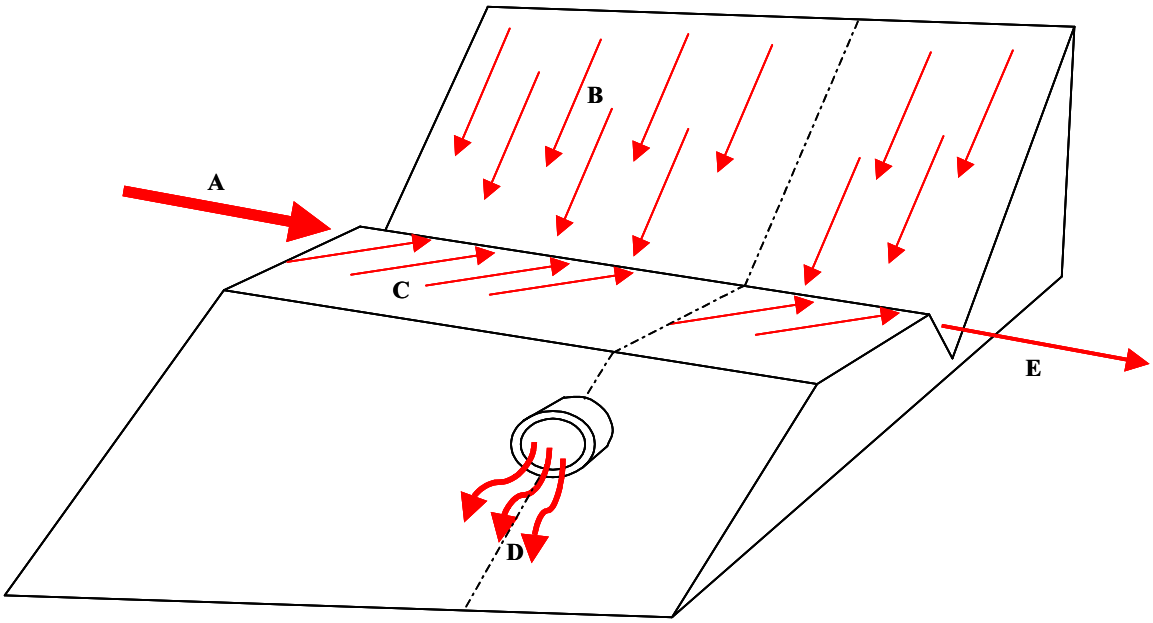


Figure 3: Cross section of insloped road prism with cross drain; A – sediment accumulated along the ditch from uphill sections; B – sediment generated by the cut slope; C – sediment generated by the road thread; D – sediment dispersed by cross drain; E – new sediment accumulation cycle.

The filtering capabilities of a side slopes vary with location as micro-topography and vegetation conditions fluctuate (*Section: Sediment Production and Delivery Mechanisms*). As sediment accumulates continuously along the stretches of road between culverts, the amount available at each location is directly affected by the cross drain spacing. An effective reduction of the sediment delivery from a road network requires a certain number of culverts strategically placed to take advantage of local filtering capabilities (Figure 4). It is important to note that these locations may not always coincide with optimal locations for prism drainage but are not mutually exclusive. When the amount of sediment available for delivery at a particular culvert location exceeds the lateral slope's filtering potential sediment could still reach the stream. These cases are often met where a road is located too close to the valley bottom or a cross drain is placed too close to a stream crossing.

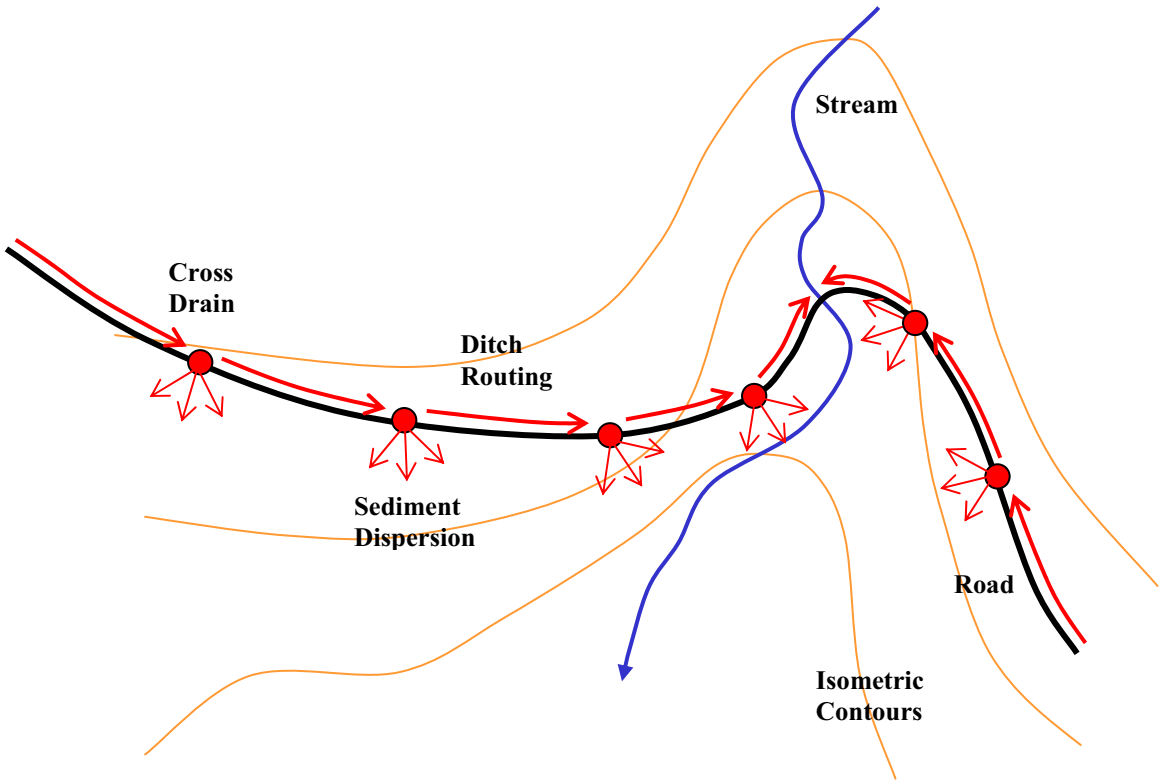


Figure 4: Cross drain system dispersing sediment. The red arrows indicate sediment routing direction

Policy and Design Restrictions

The recent aquatic wildlife crisis in the Pacific Northwest in conjunction with an increased awareness of environmental issues led to the adoption of a new policy for the timber industry. As forest roads are considered potentially harmful to the environment a particular set of restrictions has been imposed on forest road construction that directly affects the cross drain system design. Designers are required to minimize entry of ditch water and surface sediment into streams by dispersing sediment onto the forest floor.

Cross drain spacing is seen as an essential aspect of sedimentation reduction. Generally, cross drains are to be spaced at regular intervals along the road. The distance between culverts is provided as a function of road grade, side slope, average distance above streams, road surface condition and use, precipitation, and soil erosion potential. The

rules also specify that the distance between a stream crossing and the first upslope cross drain is important to the volume of sediment delivered and recommend that a culvert should be installed 50 to 100 feet above all stream crossings (*WA Forest Practices Board Manual 2000*).

Even though the site filtering potential is not explicitly referenced by the current regulation, the cases where culverts are too close to a stream crossing and diverted sediment could still reach the stream network are recognized. The manual recommends either avoidance of such situations or implementation of additional measures such as sediment traps or ponds, rock armored ditches, and vegetated ditches.

A sufficient number of cross drains should be installed in order to prevent ditch scour, over flowing cross drain capacity or erosion at cross drain outlets. The manual requires designers to make use of natural swales that the road crosses in order to avoid rerouting water along the ditch, where it can pick up and transport sediment.

Using these guidelines makes it possible for an experienced professional to design a functional cross drain system. However, one potential draw back of spacing cross drains at regular intervals is that the site filtering potential might not be fully exploited. In areas highly susceptible to sedimentation, a non-uniform spacing of culverts to take advantage of local terrain and lateral sediment retention capabilities might be more suitable. For a more rigorous analysis of the sediment production and delivery from forest roads some specific software tools exist.

Sediment Modeling and Existing Tools

Due to the complex nature of the sedimentation process a precise measurement of sediment impacts form forest roads is not always possible. Several software programs were created in order to model the road-stream sediment interaction and estimate the amount of sediment delivered.

The Water Erosion Prediction Project (WEPP) is a simulation program, originally developed for agricultural purposes as a replacement for the Universal Soil Loss Equation (*Elliot et al. 1999b*). It is a complex program that models the processes that lead to erosion including infiltration and runoff, soil detachment, transport and deposition, plant growth and residue decomposition. WEPP works with a given slope profile and runs simulations over a specified period of time under a multitude of customizable parameters. The Forest Service Moscow Lab has developed a set of specialized interfaces in order to simplify WEPP use for erosion and sediment delivery from forest roads.

X-DRAIN is a basic interface to accessing predicted sediment yields from over 130,000 WEPP simulations ran by soil erosion specialists (*Elliot et al. 1999a*). It was developed to simplify and speed up the application of WEPP for simple forest road settings. The end user has limited control over climate, soil, side slope and distance to streams parameters. Road geometry and distance to streams are assumed uniform along the analyzed road segment. The total sediment yield in lb/year is presented on a tabular form for a fixed number of combinations of road gradient and cross drain spacing values.

WEPP:Road is meant to be a more refined sediment modeler capable of modeling one road segment at the time. As inputs, it accepts road surface information and a customizable climate description. The modeling can be done over a user defined period of time (Figure 5). The sediment yield to the stream network in lb is reported as a single number together with the additional average precipitation, runoff, and the amount of sediment leaving the eroding portion of the road prism. There is also the option of an abbreviated hillslope output presenting a distribution of erosion and deposition, the presence of a sediment plume in the forest, and the particle size distribution of sediment delivered to the channel (*Elliot et al. 1999c*).

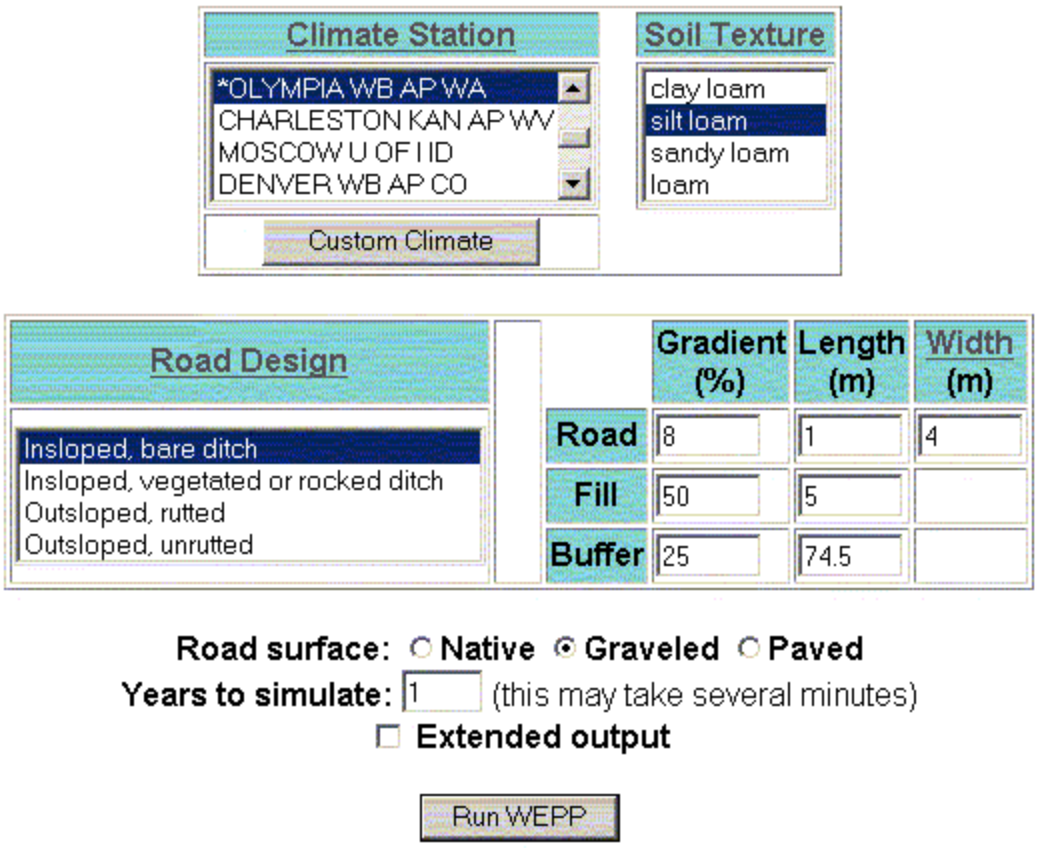


Figure 5: Screen capture of the WEPP:Road interface

One major limitation of the WEPP based programs is that they are spatially non-explicit and thus incapable of distinctly placing the sedimentation processes within a road network. Other organizations have approached this problem within a more spatially aware context using Geographic Information Systems (GIS) as a base for their analysis. GIS have become a standard in environmental modeling. Their capacity of modeling overland flow is what makes them indispensable to spatially distributed phenomena involving streams and water routing.

SEDMODL is a GIS based, road erosion and delivery model developed by Boise Cascade Corporation in cooperation with the National Council on Air and Stream Improvement.

The model identifies road segments with a high potential for delivering sediment to streams in a given watershed. It uses spatial information to determine the proximity of the roads to the stream network. Sediment delivery is then calculated for the roads that drain to streams using methods derived from the Washington Department of Natural Resources Standard Method for Conducting Watershed Analysis and WEPP. The program is designed as a flexible, multipurpose tool that can be used both for screening purposes or a more detailed sediment analysis. For more reliable results a set of specific road attributes is required. They can include: road use, surface type, road width, construction year, cutslope height, road geometry type, and road gradient. If culvert locations are known they can be inputted as GIS layer and will affect sediment computations accordingly (*National Center for Air and Stream Improvement 2002*).

The Washington Road Surface Erosion Model (WARSEM) is new software from Washington Department of Natural Resources intended as a long term road management planning tool. It can model sedimentation and drainage at four different levels from the broad basin scope to the individual road segment. An increasingly complex amount of road information is required with each superior level. The model stresses out the importance of accurate, field verified input data towards a successful sediment budget. The model is implemented as an Access database without a spatial component. SEDMODEL2 results can be imported to generate performance metrics in a long term best management practices analysis (*Watershed GeoDynamic et al. 2003*)

The Need for a Specialized Cross Drain Design Tool

The contemporary emphasis on modeling the processes that lead to erosion and sediment transport resulted in a well represented collection of computer programs. These software tools were created as analysis packages and perform well when used for general identification of sedimentation problem areas or when examining isolated parts of already built road networks. However, as multiple design alternatives are evaluated during the road design stage, these tools may fall short of functionality, prove slow and relatively

ineffective. They are focused on accurate modeling of sedimentation processes but do not provide the means for minimizing total sediment delivered, especially important when designing cross drain systems. None of these tools actively address the question of how culvert placement impacts sediment delivery to stream networks. For example, as WEPP:Road works with only one road segment at a time, using it to determine best culvert locations and optimal spacing of a road drainage system would require multiple simulations for each potential culvert movement. A typical investigation involving many case scenarios, with culverts placed at various locations, requires repeated runs of the model for all road segments affected by a the presence of a cross drain, at each particular snap shot. This could become a very inefficient, time-consuming procedure especially for long roads with multiple cross-drains where sediment producing factors vary a lot. XDRAIN on the other hand, can only be used for roads with uniform conditions, being limited to a constant road grade and considering culverts to be uniformly spaced. Users cannot tell which one of the culverts has more potential to deliver sediment and it is impossible to know what placement would possibly reduce sediment impact to streams. SEDMODL automatically identifies and displays road segments that are probable to deliver sediment based on, among other factors, a GIS layer of known culvert locations. If trying to find the best possible locations for existing culverts or revise the number of culverts to reduce sediment impacts, the culvert layer has to be modified manually and the model rerun. This process, as in the WEPP case, can become inefficient if repeated many times.

There is presently no methodology that allows a road engineer to quantify the effects of varying culvert spacing or selecting specific culvert locations, on sediment delivery to streams. The absence of such a design tool makes it more difficult to take culvert spacing into account as an effective solution for reducing sediment impacts from forest roads.

Goals and Objectives

Goal:

- Investigate culvert placement as a method for reducing sediment delivery to stream networks from forest roads.

Objectives:

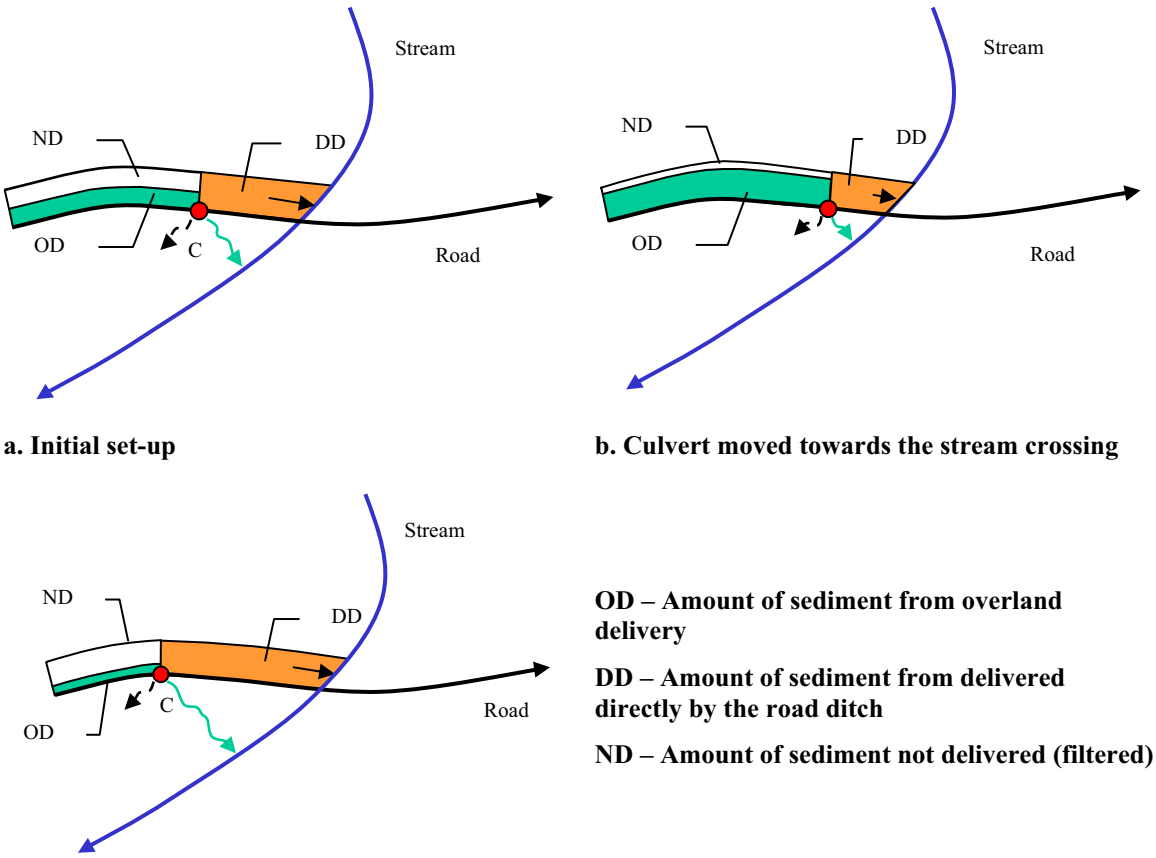
- Develop a software program that allows engineers to quantify the effects of varying culvert location along a road network with the following properties:
    - Spatially explicit – associates sedimentation to precise locations on along the road.
    - Integrated with a standard geographic information system platform.
    - Efficient - evaluates alternatives in a timely manner without using external programs of procedures.
    - Ease to use.
- Illustrate the effectiveness of culvert placement for reducing sediment delivery to streams with a real road-setting example.

Concepts

The Cut-Off Culvert

In order to analyze cross drain systems design and minimization of sediment delivery from forest roads we introduce the term *Cut-Off Culvert*. The cut-off culvert is any culvert that directs water across a vegetated hillslope. This differs from standard drainage culverts that divert water into a stream channel. Typically cutoff culverts are placed solely to reduce ditch erosion and maintain ditch flow capacity. Concurrently they can also be used to reduce the direct delivery of sediment-laden water from the road's ditch by diverting it onto the forest floor where a major part of the sediment will be filtered out and retained prior to entering the stream (*Section: Cross Drain Systems and Sediment Reduction*). From the sediment reduction standpoint, the cut-off culvert's location is crucial to the well functioning of a cross drain system. Figure 6 shows the effects of placing a cut-off culvert at different locations along a road segment.

The volumes of overland delivery (OD) from culvert C after filtering and direct delivery from ditch (DD) are represented by the two colored areas. The total sediment delivered is given by the summation of these two areas and fluctuates with culvert placement. Assume that the amount of sediment produced by the road prism is constant in all three cases, the side slope has uniform filtering capabilities and the lateral filtering is proportional with the distance the sediment must travel downhill this side slope. It can be noticed that by placing culvert C closer to the stream intersection increases its lateral delivery potential while it decreases the direct delivery from ditch (Figure 6b). Conversely moving culvert C away from the intersection gives more filtering power but also leaves more contributing area for direct delivery (Figure 6c). The question of ideal placement of a cut-off culvert becomes a question of maximizing the filtered sediment (ND) at the expense of direct delivery (DD) and overland delivery (OD)



a. Initial set-up

b. Culvert moved towards the stream crossing

c. Culvert moved away from stream crossing

Figure 6: Effects of moving cut-off culvert C along the road alignment. To minimize sediment delivery the sum of overland delivery OD and direct delivery DD has to be minimized by increasing filtering ND.

Minimization of Sediment Delivery for a Simple Case

To explore the optimal location of a single cut-off culvert a simplified case scenario was built (Figure 7). A straight 150m long road segment, of constant grade, crossing a uniform slope hillside intersects a stream at an angle α. The sediment generating factors along the road segment and the filtering characteristics of the side slope are invariable. A cut-off culvert is placed at a random location along the road alignment. A direct delivery distance through the ditch (d) and a sediment- filtering potential characterize every possible location.

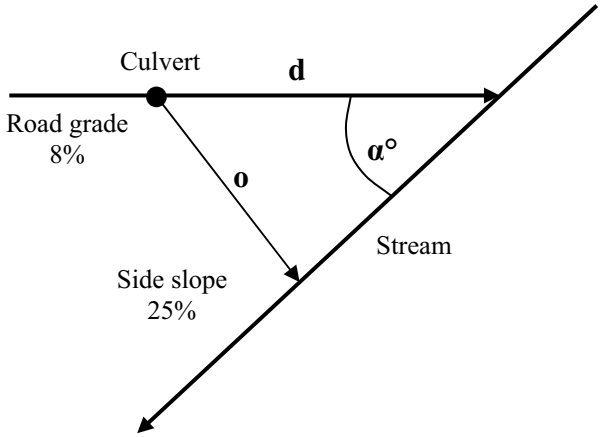


Figure 7: Analysis setup for a single cut-off culvert scenario

The water dispersed by the culvert in question usually travels towards the stream on a sinuous path following the local topography (*flow distance*). For simplification purposes assume that the side slope's topography is uniformly flat and this flow path is identical to a straight perpendicular to the stream (*Euclidian distance*). The generic term: overland distance (*o*) refers here to the Euclidian distance between culvert and stream (Figure 7).

The sediment filtering potential expressed here as the proportion of total available sediment delivered to stream (*F*), varies with the overland distance (*o*) as well as other site-specific factors like: soil types, local gradient and vegetation cover. Because the overland distance (*o*) is in this case proportional with the direct distance (*d*) it can be stated that the filtering potential (*F*) is also proportional with the direct distance (*d*). By expressing *F* as a function of *d* the total sediment delivered to the stream can be budgeted with Equation 1:

$$T = K \cdot d + K \cdot \left(L - d\right) \cdot F(d)$$

Equation 1: Total sediment as a function of culvert location *d*

T = total sediment delivered (kg)

K = sediment production rate (kg/unit length of road prism)

L = total road length

F = fraction of sediment delivered (non-dimensional)

The minimum delivery T is obtained when:

$$\frac{\partial T}{\partial dd} = 0 \Rightarrow 1 - F(d) + (L - d)F'(d) = 0$$

Equation 2: Optimal culvert location *d*

The optimal location for cut-off culvert C can be obtained by solving differential Equation 2 for *d*. It can be noticed that the optimal location of C is not dependent on the sediment production rate K but only on the road-stream geometry and the local filtering characteristics.

In actuality, absolute deterministic models of side slope filtering potential to take into account every combination of factors encountered on forested terrain are not currently available. As the fraction of sediment delivered *F* cannot easily be determined a completely analytical approach to culvert optimization is not feasible.
Various researchers have approached the problem on empirical bases, using statistical inference on their field measurements in order to model sediment delivery. Equation 3 describes the volume of sediment deposition as a function of the travel distance for particular conditions encountered in the Idaho Batholith (*Ketcheson and Megahan 1996*).

$$F = \left(103.62 \cdot e^{\left(\frac{-x}{32.88}\right)} - 5.55 \right) \cdot \frac{1}{100}$$

$$x = \frac{o}{o_{\max}} \cdot 100$$

Equation 3: Empirically derived fraction of sediment delivered expressed as the percent of the available sediment volume; *o max* is characteristic to granitic waterseds in Idaho Batholith (Ketcheson and Megahan 1996)

By plugging equation 4 into the sediment budget equation the total sediment delivered for our case scenario can be plotted as a function of the culvert location (Figure 8). A family of curves is shown for various sediment production rates. A minimum delivery is obtained with culvert C at 50 meters away from the stream intersection all across these different sediment production regimes. It is important to note that empirically derived equation 4 cannot be universally applied. However, assuming that in most cases overland sediment delivery follows analogous invert exponential distributions (*Burroughs and King 1989*) it is possible to effectively approximate near optimal locations for culverts.

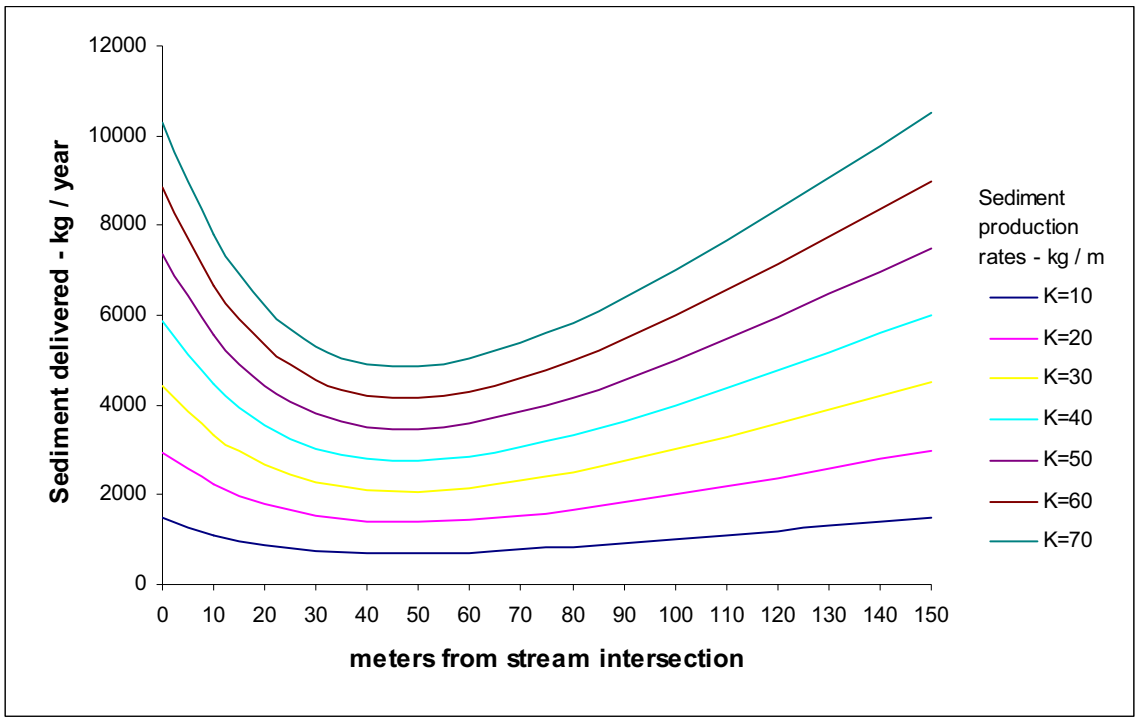


Figure 8: Sediment delivery rate to stream for various sediment production rates. Road segment is 150 m long, 4m wide; alpha is 45 degrees; sediment production rates K in kg/meter of road prism. The fraction of sediment delivered is computed with Equation 3

Similar approaches have already been incorporated into some more advanced sediment modeling programs. Therefore in order to further investigate the optimal placing of a cut-off culvert, the F.S. WEPP:Road interface was called upon to provide the modeling for the sediment production and delivery for the proposed case scenario. An experiment was

conducted where culvert C from the set-up in Figure 7 was incrementally moved along the road segment. The sediment impacts associated with each of these locations were quantified by WEPP simulations run at the Forest Service web site. The sediment production parameters were: 4m wide, insloped bare ditch road, 5m long fill slope at 50% gradient and silt loam soils. Olympia Station described the local climate. The forested buffer had a uniform 25% slope. All simulations were performed for a 1-year period. Multiple runs were completed for various stream crossing angles α. Figure 9 presents a graph of the results of this experiment. The total sediment delivered exhibits a similar behavior to the Ketcheson and Megahan approach presented in Figure 8. It follows a right skewed distribution with a minimum in the first 1/3 of the road segment. The optimal culvert location for this particular case is located 50-60 m away from the stream crossing. The geometry of the road-stream intersection has a major impact on the filtering distance and implicitly affects the optimal cross drain location.

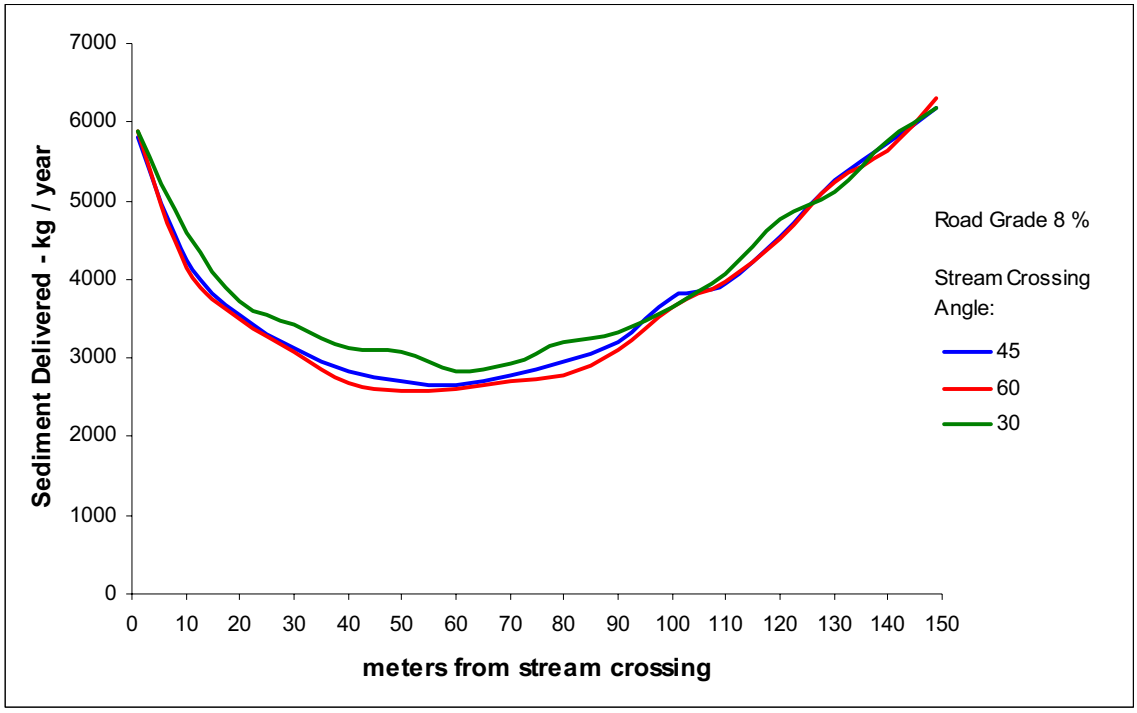


Figure 9: Sediment delivery rates for various stream crossing angles; Road grade is constant at 8 %; culvert at position on x-axis.

Exploring Optimization of Cross Drain Systems

Elaborating on the simple geometry single culvert case scenario presented above, the question of further minimization of sediment delivery with the introduction of additional culverts was explored. Another experiment was conducted where two adjacent culverts were moved away and toward each other in an attempt to obtain a reduction in sediment delivery. The setup for this experiment is seen in Figure 10. Note that the essential simplification of flow path distance being identical to the Euclidian distance was maintained from the previous case (*Section: Minimization of Sediment Delivery for a Simple Case*)

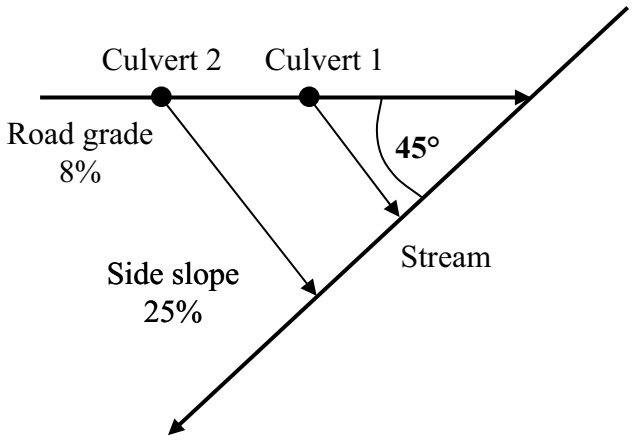


Figure 10: Setup for the two cross drain experiment

The total sediment delivered was recorded again using WEPP:Road. All WEPP sediment-modeling parameters set in the previous experiment (one culvert scenario) were kept unchanged. The stream intersection angle α was set at 45°. Starting with the cut-off Culvert1 in the optimal position previously determined at 60 m from the stream, the total sediment delivered was computed with Culvert 1 and Culvert 2 at several different locations. Due to the high number of possible combinations of 10m increments over a 150 m road segment only a few were fully explored. Table 1 present the results of this experiment. The sediment delivery to stream S was reduced by as much as 58 % with the cross drains at 30 and 90 m away from the stream crossing as compared to the case of a

single cut-off culvert in optimal location. Other possible combinations may further improve this result. It can be inferred that the addition of a third culvert at a key location could decrease sedimentation even more.

Table 1: Reduction of total sediment delivered (overland and ditch) with 2 culverts; road is 150m long, 4 m wide; stream crossing angle is 45 degrees. Configuration 12, with Culvert 1 at 30 m and Culvert 2 at 90 m from stream crossing delivers the least sediment.

Config. No.	Culvert 1 location (m)	Culvert 2 location (m)	Sediment Delivered (kg / yr)	Sediment Reduction (%)
0	60	-	2661	0
1	70	100	1471	45
2	70	110	1497	44
3	60	100	1284	52
4	60	110	1349	49
5	60	90	1290	51
6	50	100	1217	54
7	50	90	1227	54
8	40	100	1125	58
9	40	90	1140	57
10	40	80	1219	54
11	30	100	1165	56
12	30	90	1112	58
13	30	80	1132	57

Full optimization of cross drain systems becomes more complex with the addition of each new culvert but the importance of the cut-off culverts closest to the stream crossing cannot be overstated. In reality, departures from the simple triangular geometry assumed here, further complicates this analysis. As road alignments and stream paths curve and

weave randomly, the road's proximity to the stream and implicitly the side slope's filtering potential become more difficult to model analytically (Figure 11). The flow path of the water dispersed by the cross drain follows the local micro-topography and can be significantly different from the straight line distance between culvert and valley bottom. Moreover sediment generating factors, vegetation and filtering characteristics vary along the road alignment and side slope respectively adding further complexity to sediment delivery modeling.

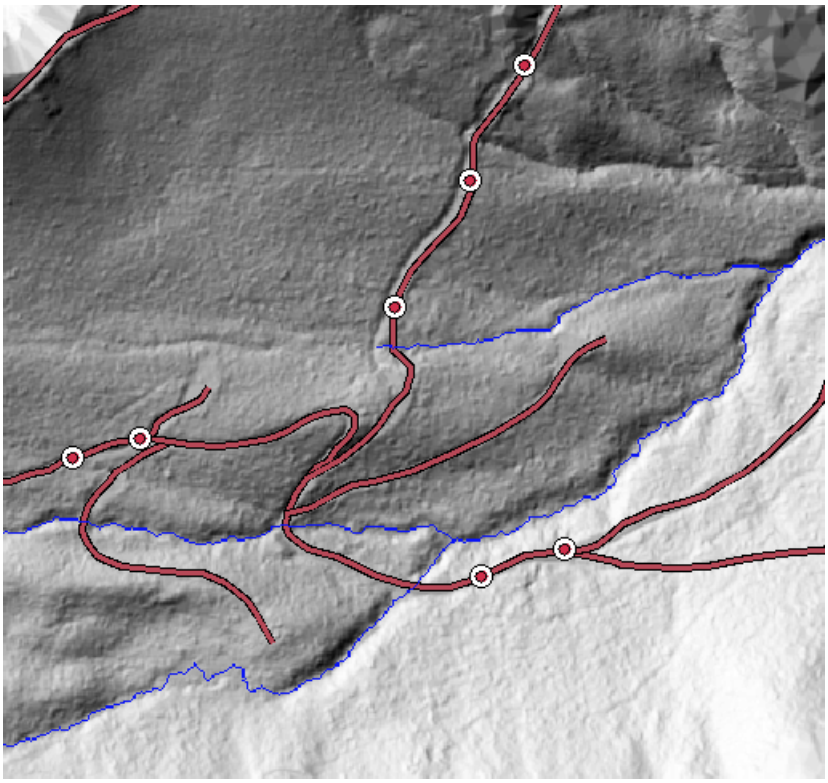


Figure 11: Example of a real case road layout with culverts

In addition to their stream crossings areas, roads are also susceptible to sediment delivery in all cases where they run parallel, in close proximity to streams (e.g. valley bottom roads). Whenever the volume of sediment diverted onto the side slope exceeds the local filtering potential, sediment delivery will occur. The process of finding best culvert

locations in these situations follows similar analysis principles with the evident omission of the direct delivery from ditch component.

Computer Modeling of Cross Drain Systems

Culvert Location Analysis for Design Purposes

Optimizing cross drain culvert spacing takes into account a composite sum of factors including terrain information, road layout, and existing culvert locations making it difficult to be performed by hand. As automated optimization software packages do not currently exist designers are limited to using various sediment analysis programs in order to estimate the validity of their design. The investigation of best cut-off culvert location using such analysis packages can become cumbersome and time consuming for longer roads with multiple stream crossings laid across non-uniform terrain. For example applying WEPP:Road for a such project requires division of the road network into segments of uniform characteristics. A culvert placed on one of these segments further divides the segment into two parts: upstream and downstream from it, each necessitating a separate run of the model. The forested lateral buffer distance at the current culvert location has to be determined. The simulation is run for each segment and the total sediment delivered to stream is computed. If one or more culverts are relocated new divisions of the original road segments and measurements of the associated buffer length are necessary. WEPP:Road is run over the Internet. Simulating sedimentation over short periods of time is a relatively quick process taking approximately 1-2 seconds per road segment. However the program can only work with one segment at a time. The manual updating of culvert positions and dynamic segmentation of the road segments are very inefficient and represent major drawbacks of this approach. Furthermore, treating road segments as stand alone entities excludes any interaction from neighboring segments. Ditch water cannot be easily carried along different segments as encountered in reality. The complexity of such analysis restricts its usage mostly to research projects. The lack of an appropriate, easy to use, culvert location analysis tool deters professional road designers and engineers from widely devising cross drain systems for sediment reduction purposes (*Schiess, personal communication*).

Interactive Culvert Placement

From the perspective of cross drain culvert design, an ideal design tool would evaluate each design step as it is proposed such that a user could easily tell the effect of his/hers decision and improve upon it. The term *interactive culvert placement* is used here to designate the process of designing cross drain systems by placing culverts at various locations with the immediate support of a sediment modeler to evaluate potential impacts to the nearby streams. Decision support tools that instantly quantify proposed alternatives typically assist this kind of interactive design. Figure 12 depicts the workflow of a cross drain design process using both an analysis package and a decision support tool.

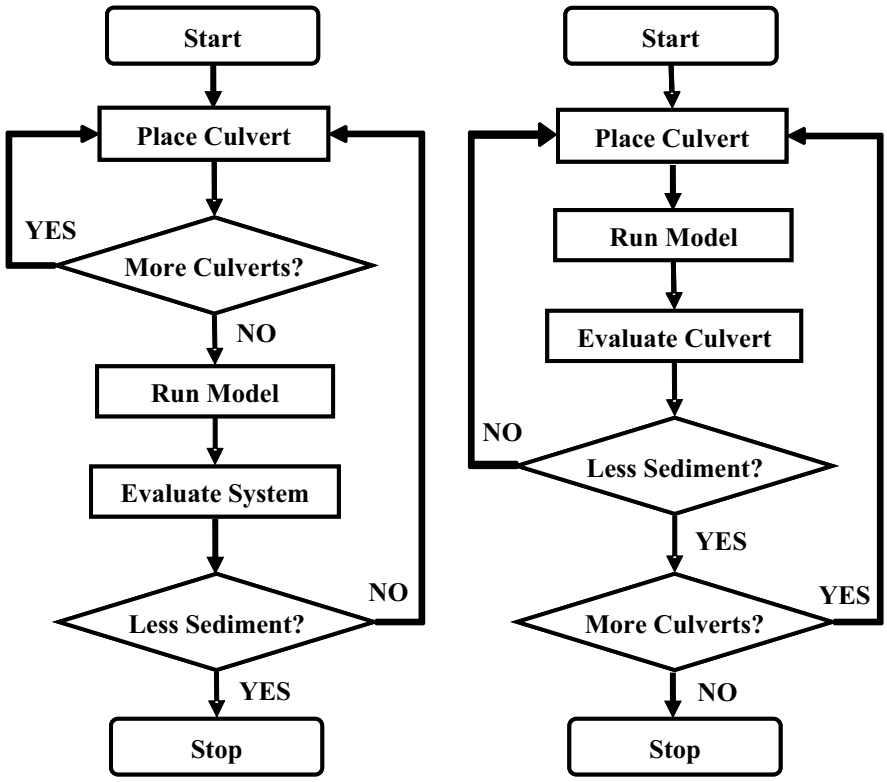


Figure 12: Flow chart of cross drain design workflow; analysis tool (left), decision support tool (right).

The major difference between the two cases is given by the feedback the user receives during design, as a validation of a placement decision. This feedback is essential to producing a good solution as users can quickly improve upon each proposed step and get closer to an optimum cross drain setup.

CULSED- A Decision Support Tool for Cross Drain System Design

To facilitate the investigation of sedimentation reduction from insloped forest roads in realistic road design circumstances, a specialized computer program named "Culvert Locator for Sediment Reduction" (CULSED), was developed. The program is a GIS based decision support tool focused on assisting engineers during cross drain system design. Its primary function is to assess the sediment delivery at each culvert location and graphically display it on the computer screen such as users can immediately ascertain the validity of a design decision. Using this program it is possible to identify near-optimal cross drain locations by exploring various permutations along the road alignment. CULSED gives users the ability to add, move and remove cross-drain culverts, dynamically evaluating the total sediment impact to the stream network from the analyzed road system. Culverts can be represented with graduated symbols proportional to their sediment delivery (Figure 13). The question of minimizing sediment delivery is thus transposed to a question of minimizing symbols on screen. The total sediment delivered by the road is displayed at all times during the analysis stages and changes with every modification performed to the cross drain system. The program has no automated procedures for achieving absolute minimization of total delivery but provides the users with the means to compare culvert locations and identify good solutions.

CULSED is implemented as an ArcGIS extension that seamlessly integrates with the standard ArcGIS package being able to access all the existing functionality and providing a familiar interface and ease of use. Running CULSED requires at the minimum a GIS road layer, a stream layer, and a digital elevation model. Additional information such as:

road surface, road age, road grade, soil and parent material and side slope vegetation cover may also be used during the sediment modeling stages. Appendix A contains a more detailed explanation of this software's capabilities.

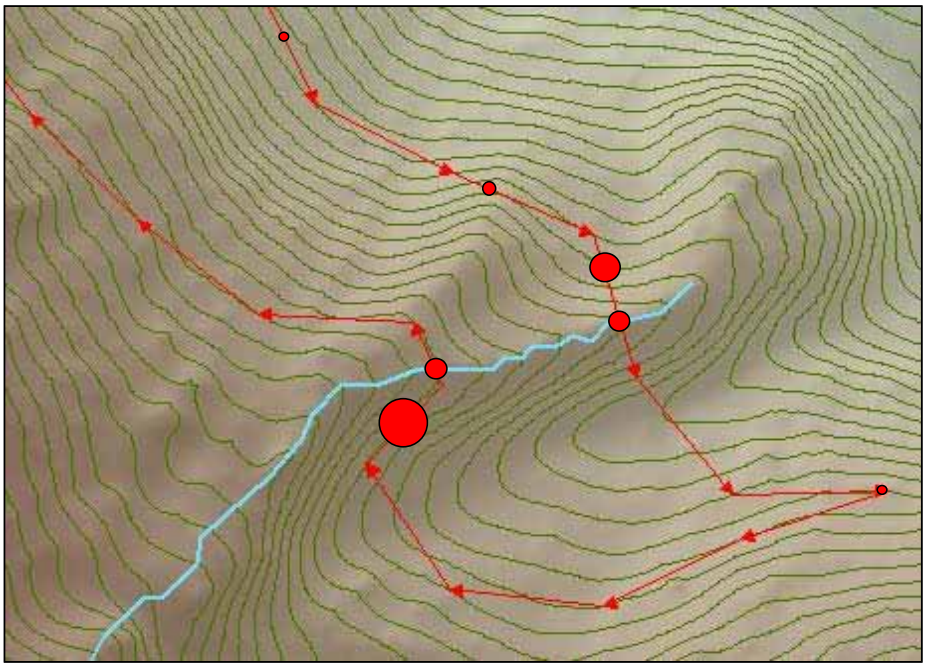


Figure 13: Sediment delivery represented as proportional symbols; a minimum size symbol represents 0 sediment delivery; arrows indicate the direction the sediment flows along the road alignment (ditch).

Modeling Cross Drain Systems with CULSED

The implementation of CULSED follows the well known Model-View-Controller paradigm. The various modules composing this architecture actively interact with each other at run time to compute and display the sediment impact related to each user action of adding, removing or moving culverts (Figure 14).

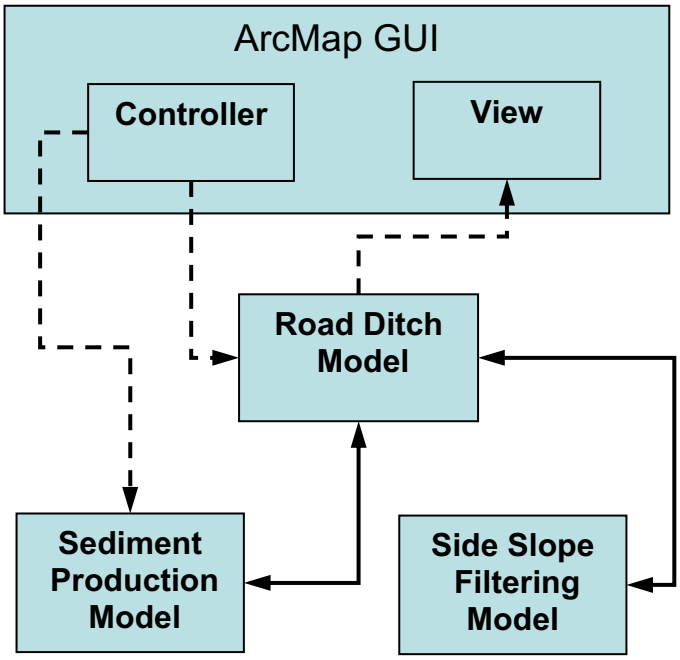


Figure 14: CULSED Model-View-Controller internal architecture; dotted lines represents implicit information flow provided by the ArcMap interface; solid lines represent explicit information exchange between CULSED's modules.

The controller module's role is to provide user input to various operations related to the culvert analysis. It essentially drives the activities of other modules that require user interaction such as: generation of network topology, cross drain movement and description of sediment producing factors. The controller module is a part of the ArcMap user interface on which CULSED is developed.

The view module is also part of this interface and is represented by the standard ArcMap graphical output. It is at this level where the results are presented to the user for evaluation and decision support. This module receives information from the ditch model, dynamically rendering all changes made by the end user.

At the core of CULSED is a suite of three modeler modules: the Road Ditch Model, Sediment Production Model and Side Slope Filtering Model. The Road Ditch Model is a

simplified representation of the road's drain ditch, modeling the flow of water along it. This module makes the following assumptions:

- All roads are insloped
- There is continuous ditch along every road segment in the network
- Water flows along the ditch and spills out only at culverts and ditch ends
- All culverts are functioning within designed parameters
- There are no intersections of more than 4 roads

The road ditch is described by the interplay of a geometric network and its associated logical network. The geometric network stores the physical location and geometry of the roads alignments, stream crossings and cross drain culverts together with their geometric connectivity. These network components are the elements that the end user sees and interacts with on the computer screen.

The logical network portrays the sediment flow within the system, storing the directionality of water movement. It is conceptually similar to a generic graph, being composed of interconnected edges and nodes. There is a direct correspondence between the elements of the logical network and the geometric network. An edge is associated here to a one dimensional stretch of road of uniform sediment producing characteristics. Edges are sources of the sediment that flows along them. A node is the abstract representation of a connection point between edges. Culverts are always node points and play the role of sinks within this architecture. Sinks capture all physical flow routed to them. From a topological perspective each edge can be characterized by parents, children, sinks and a flow direction (Figure 15). A parent is a connected ditch segment located directly upstream on the flow path, routing its sediment-laden water into the current segment. For implementation reasons the maximum number of parents is restricted to 4. Similarly a child is a connected ditch segment downstream along the flow path, receiving sediment-laden water from the current segment. Because in reality ditch water is never split onto multiple roads at intersections, the model only allows one child per segment.

The presence of a sink at the end of an edge implies no children connectivity as in reality a cross drain intercepts and re-routs all the water it captures.

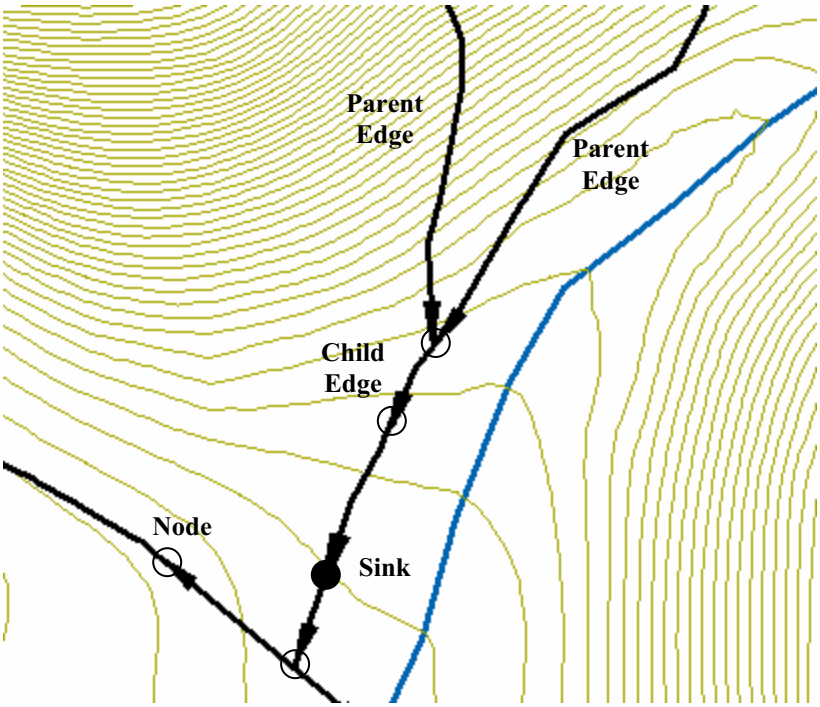


Figure 15: Logical network of the ditch model

The Sediment Production Module used by CULSED follows the methodology outlined in the Washington Department of Natural Resources Standard Method for Conducting Watershed Analysis to compute the amount of sediment generated by the road prism (Appendix A). However, in order to provide the means for other user-implemented sediment modelers to be used with the program, this module was developed on top of a generic sediment production framework. The framework enforces a certain common functionality needed by various methodologies to work together with the other program components (Appendix B). Its main function is to calculate the sediment produced by the road prism on a segment by segment basis according to their specific sediment producing parameters. These parameters have to be associated to each road segment prior to running the analysis by the end-users unless default parameters are used.

The third module in the modeler suite, the side slope filtering module, determines the sediment filtering potential associated with each potential culvert location in road the network. It is based on the work of Ketcheson and Megahan of USDA Forest Service describing the sediment deposition on a vegetated side slope as a function of proximity to streams (*Ketcheson and Megahan 1996*). The module computes the proportion of sediment that can reach the stream at any given distance along a vegetated side slope (Equation 3). The proximity to the stream is based on the physical flow path distance from the culvert to the nearest stream generated from a digital elevation model.

The sediment delivered by each culvert is calculated in response to various user triggered events as the sum of the sediment produced by all contributing road segments multiplied by the filtering potential at that particular location (Equation 4).

$$SedDel = F(location) \cdot \sum_{0}^{contrib} Sed$$

Equation 4 Sediment delivered from each culvert. *F*= filtering potential

The total sediment delivered by the entire road drainage system is thus given by sum of all contributing culverts (Equation 5).

$$TotalSed = \sum_{0}^{culverts} (SedDel) = \sum_{0}^{culverts} \left[F(location) \cdot \sum_{0}^{contrib} Sed \right]$$

Equation 5 Total sediment delivered by road

Results of a Cross Drain Redesign Application

In order to proof the concept of the cut-off culvert and demonstrate its applicability in a real world case scenario, several road settings were examined within the context of a forest harvest plan. The goal of these experiments was to reduce the total sedimentation from an existing road network by redesigning its cross drain system to make better use of sediment dispersion and filtering potential on vegetated side slopes. As the cost of the resulting drainage system had not to exceed the original cost, the total number of cross drains could not be increased. CULSED was used to compute the amount of sediment delivered by the entire road network at each design alternative during the design process.

North Tahoma Planning Area

The North Tahoma State Forest is an area of approximately 25,000 acres of forested terrain on steep topography, with a well developed road network. It is located along the Nisqually River, near Ashford WA, contained within T15N, R6E and T14N, R6E. This site was chosen for our cross drain design experiment due to its fragmented terrain with many streams and road-stream crossings. A sufficient amount of site descriptive information was available. Digital datasets of existing cross drains, roads and high resolution digital elevation models were obtained from WA DNR. We focused our study on Reese Creek Watershed, a central part of the North Tahoma State Forest (Figure 16). The existing road network in our study area was approximately 42 miles long with 76 stream crossings. The road grade varied between 1 and 19% with an average of 6%. The majority of roads were the typical one lane forest road, 12 ft. width, surfaced with gravel and maintained in good condition. As the roads were relatively old, the side slopes were mostly re-vegetated. A cross drain system was already in place and contained 168 culverts placed according to the standard WA DNR regulation. The relatively large size of our study area and its high number of existing cross drains severely limited the user interaction with the dataset on the computer screen. Therefore the site was divided it into two smaller, more manageable parts, based on the local topography and road structure.

These roughly equal subdivisions were named the East and West North Tahoma respectively.

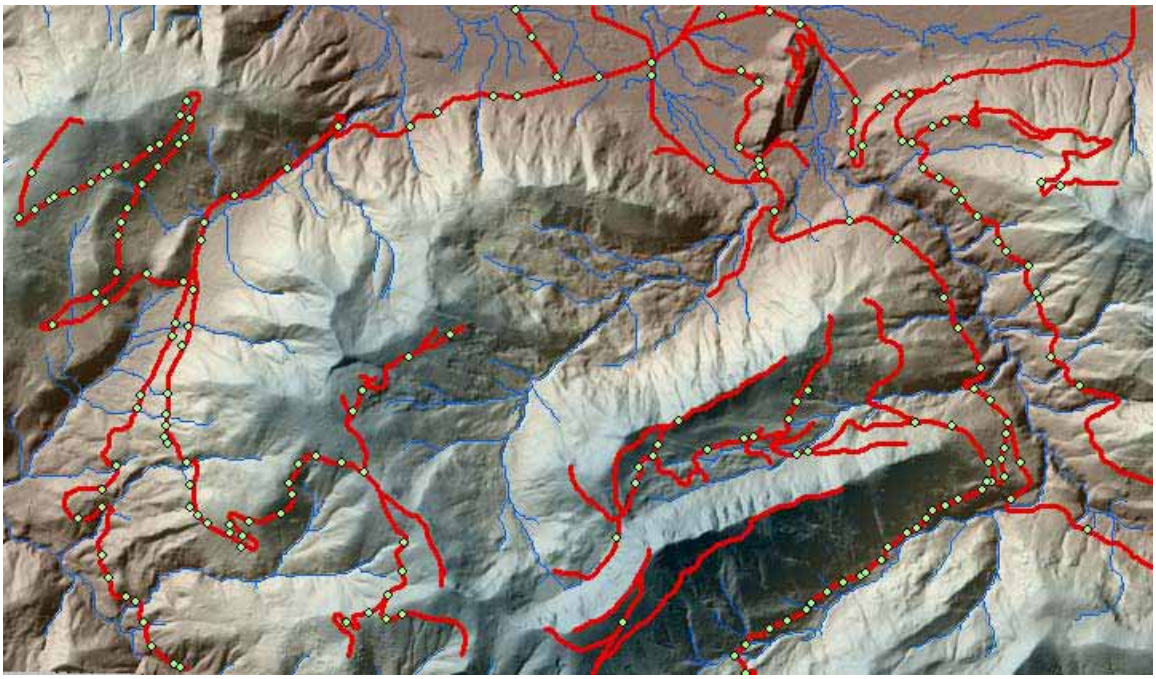


Figure 16: Shaded relief model of the North Tahoma planning area; green dots represent the original cross drain locations

Original Sediment Delivery

The North Tahoma road network within our planning area was built approximately 10 – 40 years ago. Some roads were converted from older railroad grades. Its original cross drain system had been designed to meet the specifications of the time and only minor parts had later been upgraded to current standards. Upon examination it could be observed that the cross drains had been spaced at relatively uniform intervals along the road alignments, a common practice among road engineering professionals (Figure 16). Some cross drain locations had been dictated by the terrain features, as roads went across natural wet spots, draws, swells or other formations that could lead to water saturation of the road prism and cause potential road damage. They were identified and labeled appropriately for the purpose of our analysis (Figure 17). The other cross drains had been

placed according to designer's best judgment and experience in order to reduce ditch erosion and maintain flow capacity.

All existing culvert locations were used as a starting point in our sedimentation analysis. Based on local site conditions CULSED produced an associated amount of sediment for both the East and West halves of our planning area. Figure 17 and Figure 18 show maps of the sediment delivery potential in this standard configuration, represented with proportional symbols. The total sediment delivery given by the original cross drain configuration summed up to 67.11 tons/year. This number was computed by an empirical sediment model based on the Washington DNR Manual for Conducting Watershed Analysis. Although the accuracy of this model may be debatable, it provided us with a basis for culvert location comparisons as a relative scale to measure sediment reduction.

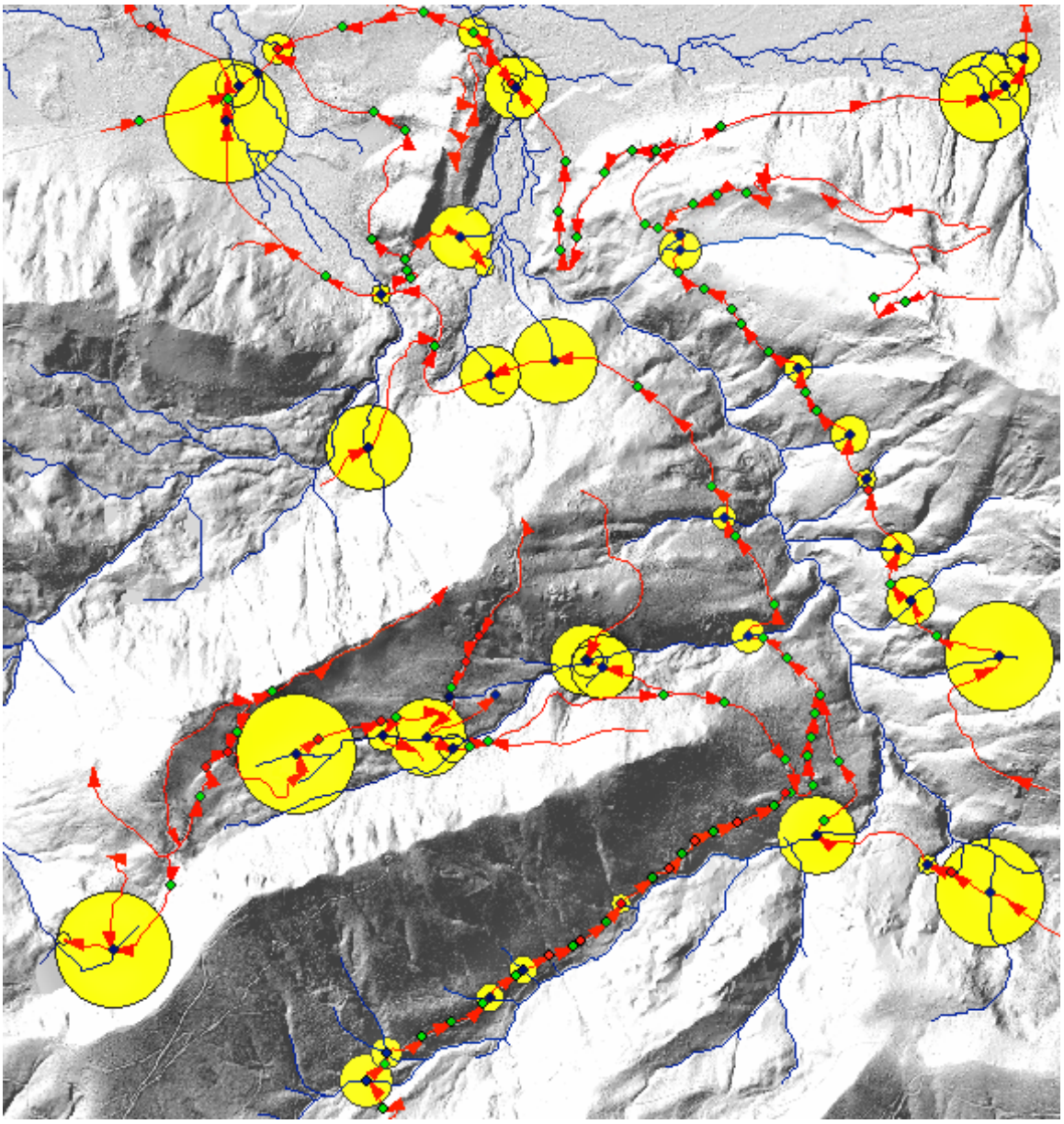


Figure 17: Initial sedimentation from East North Tahoma road network – 42.10 tons/yr; blue dots represent stream crossings; red dots represent cross drains at natural draws and wet spots; green dots represent other cross drains; red arrows indicate direction of sediment flow along roadside ditch; yellow circles are proportional to the sediment delivered at location

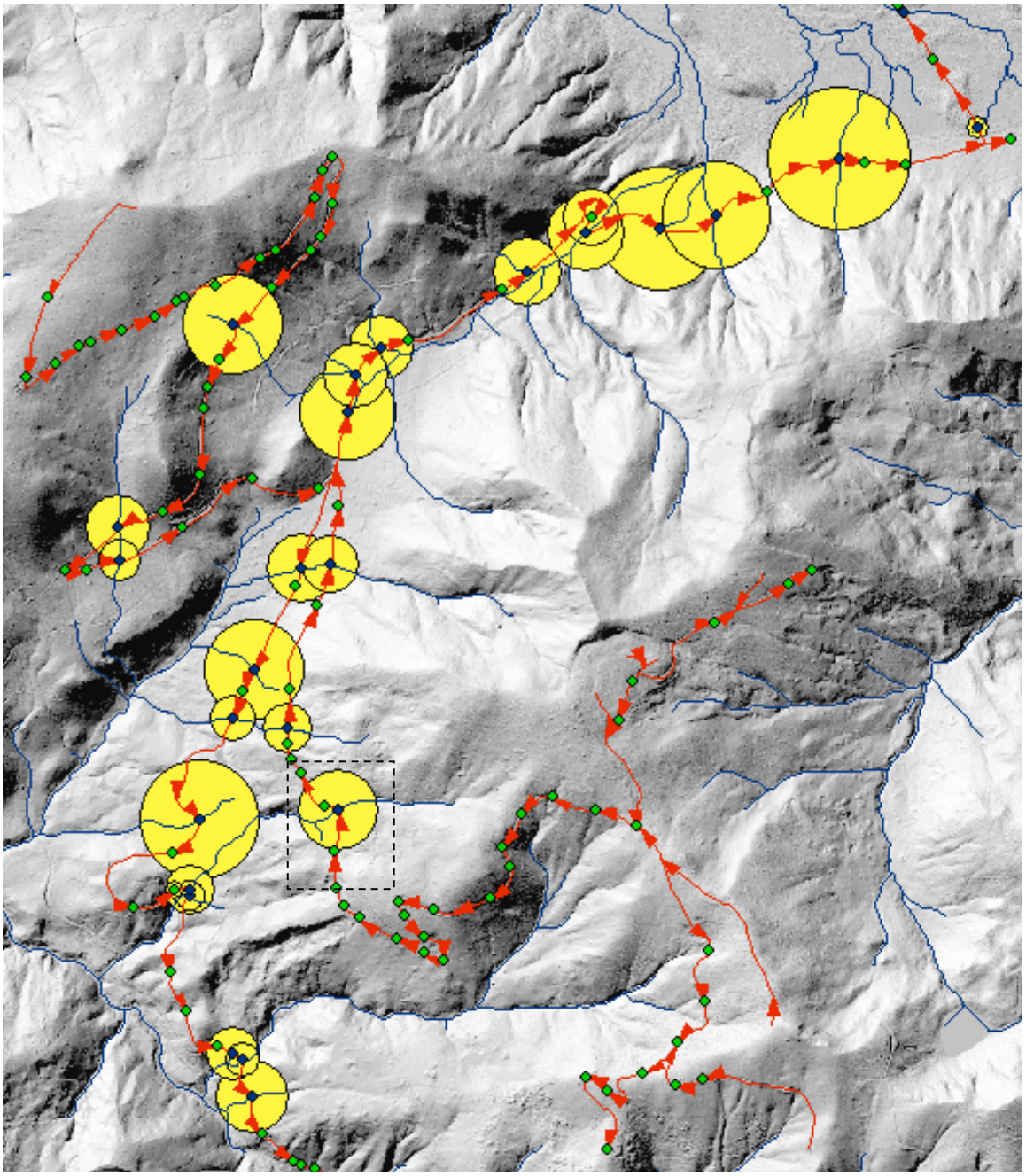


Figure 18: Initial sedimentation from West North Tahoma road network – 25.18 tons/yr; blue dots represent stream crossings; green dots represent cross drains; red arrows indicate direction of sediment flow along roadside ditch; yellow circles are proportional to the sediment delivered at location; dashed bounding box represents a subset example.

Design Process Example

To illustrate the steps taken during the investigation of a near optimal location for a cross drain culvert, a single stretch of road was analyzed in greater detail. The sample road segment, a part of the North Tahoma network, was chosen for its relatively simple set-up (Figure 18). Starting with the original culvert configuration, a cross drain was relocated progressively towards the stream crossing in five iterations. Sedimentation was graphically displayed with proportional symbols making the effects of each of these possible culvert configurations readily apparent. The goal of the road designer using CULSED was to minimize the sum of sediment delivered by all culverts involved in this operation, expressed in this case by the area of the yellow circles. Figure 19 presents the five design iterations completed in this case. The examination of sediment delivery potential at each of the five steps revealed a minimum at the third iteration with cut-off culvert placed 108 ft away from the stream crossing. This is reflected in the relative size of the graphic symbols associated with the cross drains.

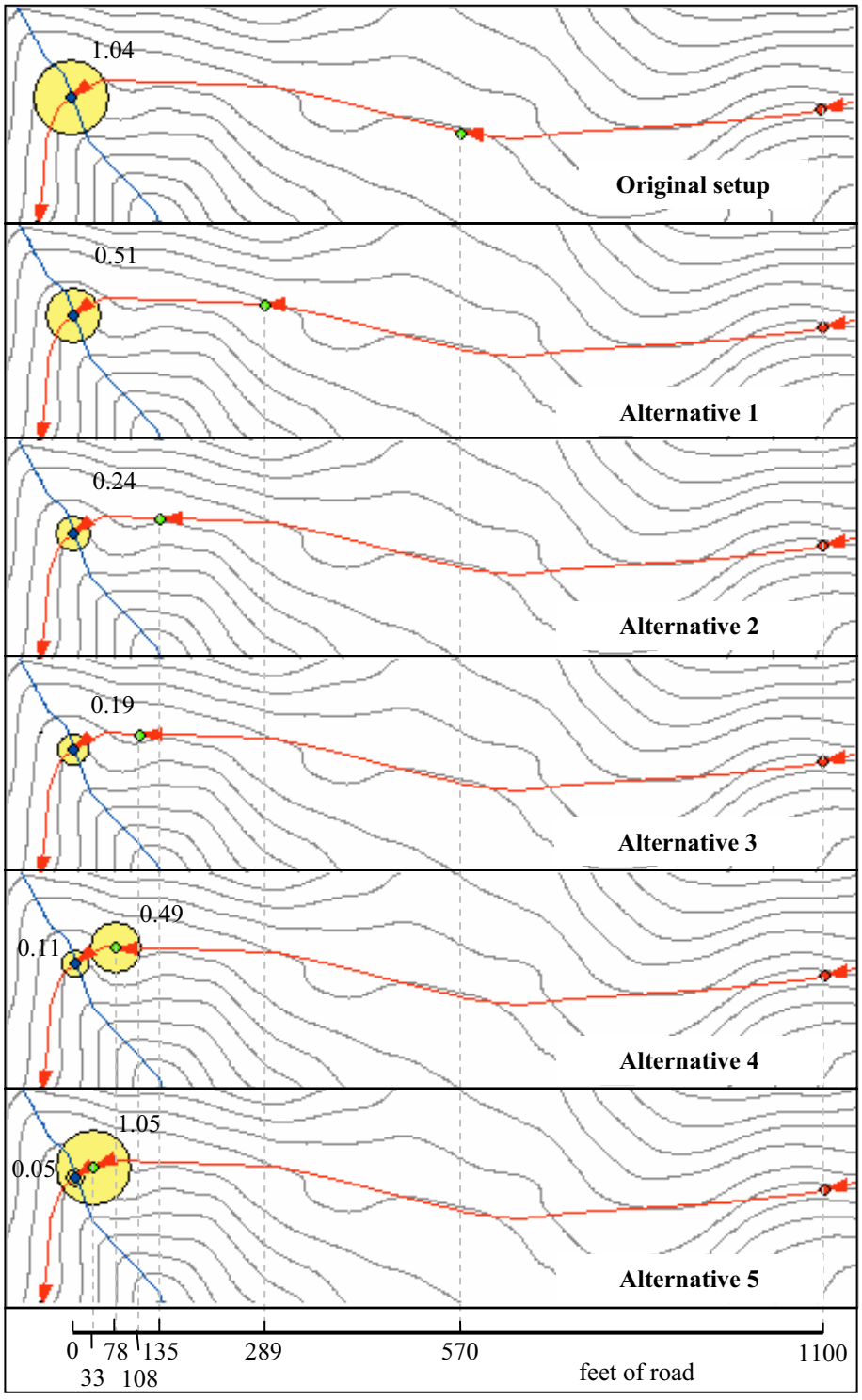


Figure 19: Composite map of a culvert placement investigation process; the circle symbols indicate the volume of sediment delivered; lowest sedimentation is obtained at alternative 3.

Sediment Reduction by Cross Drain System Redesign

By applying methods similar to the one described in the "Design Process Examples", a number of culverts were relocated to key locations in order to achieve sediment reduction for the entire North Tahoma project area. The culverts originally designed for reasons of prism health, identified and marked in the earlier steps of our design exercise, were considered "unmovable" and kept unchanged. The process of redesigning the cross drain system was attempted in stages, for both the East and West subsets of our area.
The original culvert configuration for the East half of North Tahoma consisted of 39 stream crossings, 22 unmovable drainage culverts and 64 cut-off culverts. This configuration yielded a total of 42.10 tons/year of sediment to the stream network. Acting gradually upon the greatest sediment contributors, relocating the cut-off culverts involved at each of these road settings, we were able to achieve a substantial reduction in sediment delivery. After repositioning approximately 35 cut-off culverts, the total sedimentation of the East area was reduced to 10.04 tons/year, a 76 % percent drop from the original delivery. Figure 20 displays a graphic representation of the final sedimentation. Comparing the relative size of the proportional symbols with the original configuration (Figure 17) the sediment reduction becomes obvious.

An analogous design process was conducted for the West part of North Tahoma. The original cross drain system containing 28 stream crossings, 27 drainage culverts and 55 cut-off cross drains, was potentially delivering 25.01 tons of sediment / year. Redesigning this system involved relocation of approximately 20 cross drains. The final sediment delivery was dropped to 6.33 tons/ year, achieving a 74 % reduction from the initial amount (Figure 21). The graphic quantification of this improvement can be easily noticed when contrasted with the original setup (Figure 18). The total improvement for the entire North Tahoma planning area can thus be quantified at approximately 75 % decrease in sedimentation. A number of 55 cross drains have been moved to new locations. No new culverts have been introduced in the system.

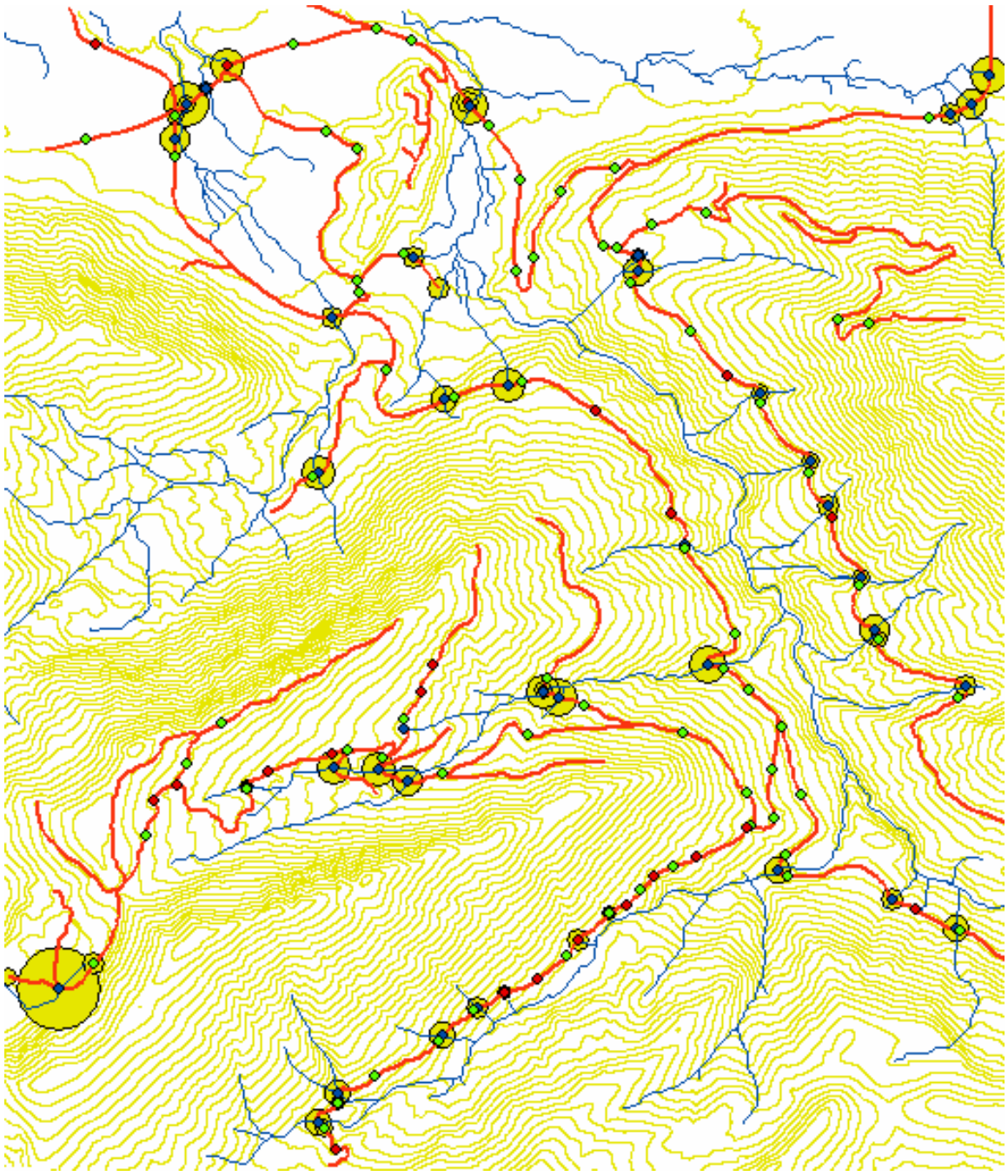


Figure 20: Sediment delivery from East North Tahoma road network after redesigning the cross drain system – 10.04 tons/ year. Yellow circles are proportional with sedimentation at the respective location; 75% sedimentation reduction from the original design.

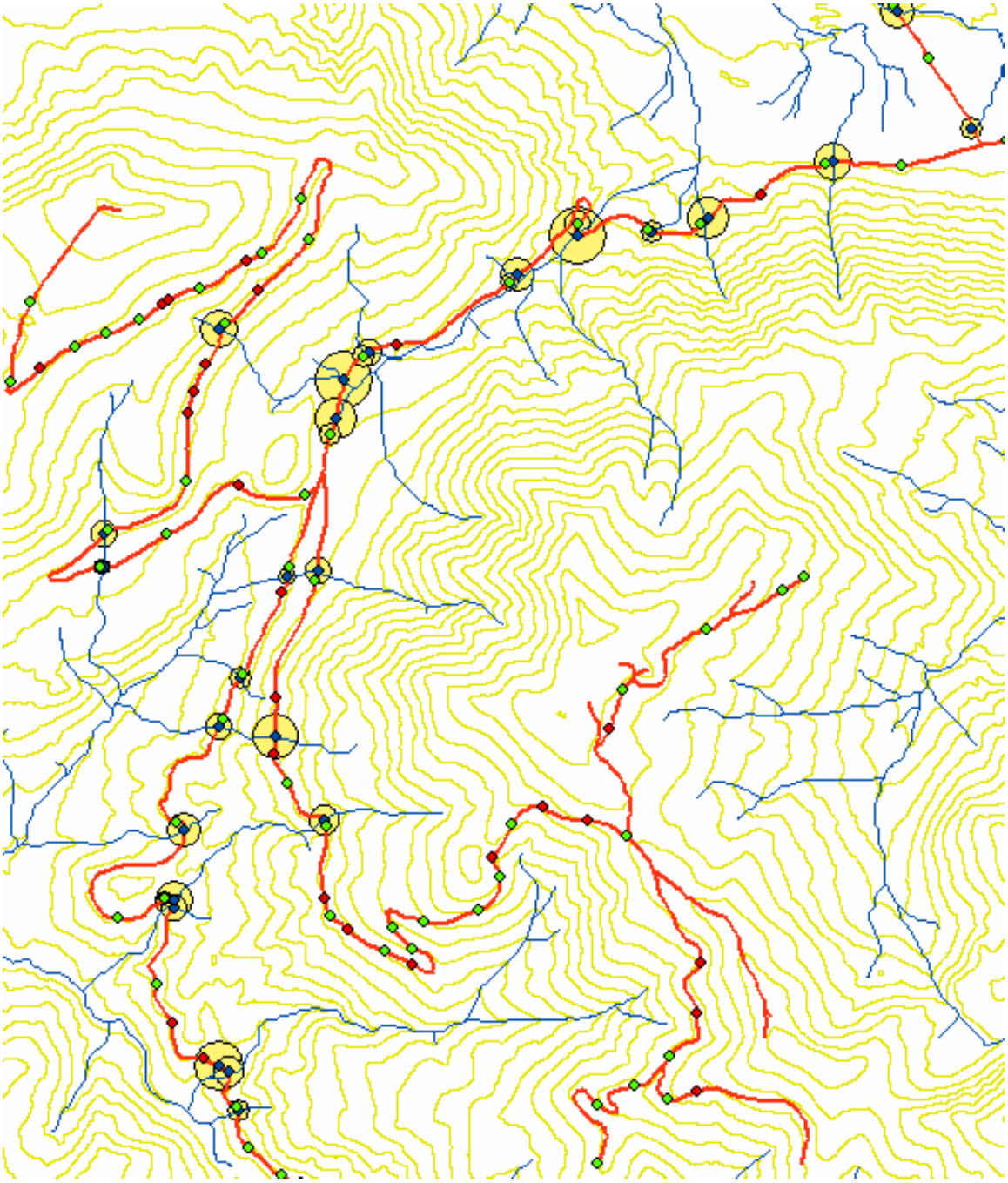


Figure 21: Sediment delivery from West North Tahoma road network after redesigning the cross drain system – 6.33 tons/yr. Yellow circles are proportional to the sediment delivered at the respective locations; 74% sedimentation reduction from the original design.

The average distance from the first cut-off culvert to the stream crossing at the end of the design process was 140 ft. A minimum of 55, maximum of 289 and a standard deviation of 59 ft were recorded. These results contrast with the Forest Practices Board recommendation of placing the first culvert within 50 – 100 ft of a stream crossing. The wider range of values produced by CULSED stems from the variability of local conditions: sediment producing factors and delivery potential, characteristic to each road location. The interplay of the direct delivery from ditch and overland delivery from cross drain is what determines the amount of sediment reaching the streams in the near vicinity of a stream crossing. Although general guidelines can be successfully applied in certain average situations, cross-drain location design is best approached on individual bases.

One important aspect to mention is the scope of the analysis carried above. Given that an absolute optimal location is impractical to obtain through experimentation (especially when dealing with a high number of culverts over a large area) a three quarter reduction of sedimentation was considered satisfactory. Further culvert manipulation could lower sedimentation even more but major improvements should not be expected.

Weaknesses and Shortcomings

The most important location for a cut-off culvert is within 100 – 200 ft uphill from a stream crossing, while other locations on the road alignment can have little to no effect on sedimentation. This is especially true on mid-slope roads situated far enough from a stream valley to benefit from full sediment dispersion and filtering on vegetated side slopes. In these conditions, designing a culvert system strictly for the purpose of sediment reduction could lead to an oversimplified drainage configuration. There is a tendency to keep the number of culverts to a minimum as they are neutral to sediment delivery. However, by spacing cross drains too coarsely, long stretches of road are left exposed to ditch scouring and infiltration of ditch water into the road structure. This may impact the maintenance costs and potentially cause road failure as the prism and/or subgrade reach

the saturation point. The road engineer's experience is crucial for the outcome of such a design project.
The benefits of the cut-off culvert are more apparent in regions with steep, fragmented topography, where higher amounts of sediment are produced and transported. Flat areas with low grade roads and few stream crossings do not gain from a complex culvert analysis. Furthermore, modeling flow patterns in these flat areas presents a challenge for the current GIS algorithms, yielding unreliable results.

A particular GIS problem affecting CULSED analysis is the modeling of the flow-path and generation of streams from a digital elevation model. The raster resolution strongly impacts the outcome of this analysis. The cell spacing determines the minimum increment of the flow path distance measurement and implicitly influences the number of valid culvert locations that can be evaluated along a road segment. Analyzing short road segments with wide cell spacing can render this technique impractical. The standard 10m and 30m DEM can only be used for smaller scale analysis with a larger tolerance for error.

Specific issues related to computer modeling of the cut-off culvert concept currently restrict its wide scale usability. CULSED was designed for insloped roads with a side ditch. The ditch model assumes ditch continuity along all roads segments. A road network containing road types where the side ditch may not be present would be misinterpreted as its sediment flows would erroneously be simulated.

Sediment production and delivery potential associated with each culvert are computed with empirical models derived for particular conditions in the Northwest of the United States (*WA Forest Practices Board 1997, Ketchesson and Megahan 1996*). These models were meant to operate on relatively large scales and adjust poorly to the micromanaging imposed by a culvert by culvert analysis. Their absolute results may not always reflect the reality of all case scenarios met in road design. Overestimation of sediment

production seemed characteristic for the North Tahoma planning area. Nevertheless, as culvert locations are evaluated on a relative scale, these figures can serve as a basis for comparison and decision support.

Discussion

Importance of Better Models

One major component of cut-off culvert modeling is the flow of surface water from precipitation. Accumulation, canalization and dispersion of this water drive processes of erosion, sediment transport, deposition and infiltration into the stream networks. Cut-off culverts together with the road ditch are directly involved in the local hydrology as man-made structures that reroute water away from its original pathways. To identify and estimate localized stream sedimentation from roads, a successful model has to be able to track the flow direction across the terrain. Flow direction constitutes the starting point of such an analysis, a foundation on which sediment can be associated with relevant locations within the study boundaries. Only after the determining the flow direction can the sediment quantification be performed. Culverts would then be spaced according to their associated amount of sediment. This level of functionality is supported by CULSED, a decision support tool for cross drain design introduced above. An important enhancement to be made to this approach is to evaluate the actual amount of water traveling along the road's side ditch. By knowing how much water is accumulating at any point on a road alignment, ditch scouring processes could be modeled. Cross drain spacing could therefore take into account scouring in order to avoid problems related to long, unprotected road segments. Moreover, the estimated amount of water flowing through a particular location could serve as a parameter for dimensioning culvert pipes. By examining an actual road set-up, existing culverts susceptible to overflow could also be identified and marked for a potential redesign.

Another aspect of the current method that could benefit from improvements is the modeling of the sediment delivery process from a cross drain to the nearest stream. A deterministic model of water infiltration through the forest soil has the potential to increase the accuracy of our sediment filtering prediction. Coupled with the average magnitude of water flows outwards from a specific culvert, such a model would quantify

sediment deposition over distance to stream. Local terrain conditions influencing this process could serve as inputs. This model would be easily incorporated into the present GIS based analysis routines which best represent spatially distributed phenomena on a fixed point in time.

A common occurrence in GIS environmental analysis also employed by CULSED is the use of raster elevation models (DEM) to describe terrain features. They are in essence a discrete pixel representation of the ground topography. In terms of pixel resolution and generating method, various DEM standards exist but their ability to capture topographic details varies. When performing a detailed analysis as required by cross-drain modeling, the topographic expression is critical to the accuracy of the results. Natural terrain features such as small stream valleys, draws, swells and other low spots influence local hydrology, road layout and implicitly sedimentation. More particularly, in a computer environment they affect the modeling of the flow path, at the base of the cross drain design process. Recently, high resolution DEM, a new standard in terrain modeling have been introduced. Their ability to reveal a lot of the micro-topography makes them suitable for cross drain analysis. The North Tahoma Redesign Project has been carried out on a 6 ft high resolution DEM. A shaded relief of this model clearly illustrates the micro-topographical detail included in that analysis (Figure 22). Although new, this kind of data is rapidly becoming available at lower costs. Organizations such as Puget Sound LIDAR Consortium are developing datasets of large extents. As this technology matures and turns more accessible, the usability of tools like CULSED will increase.

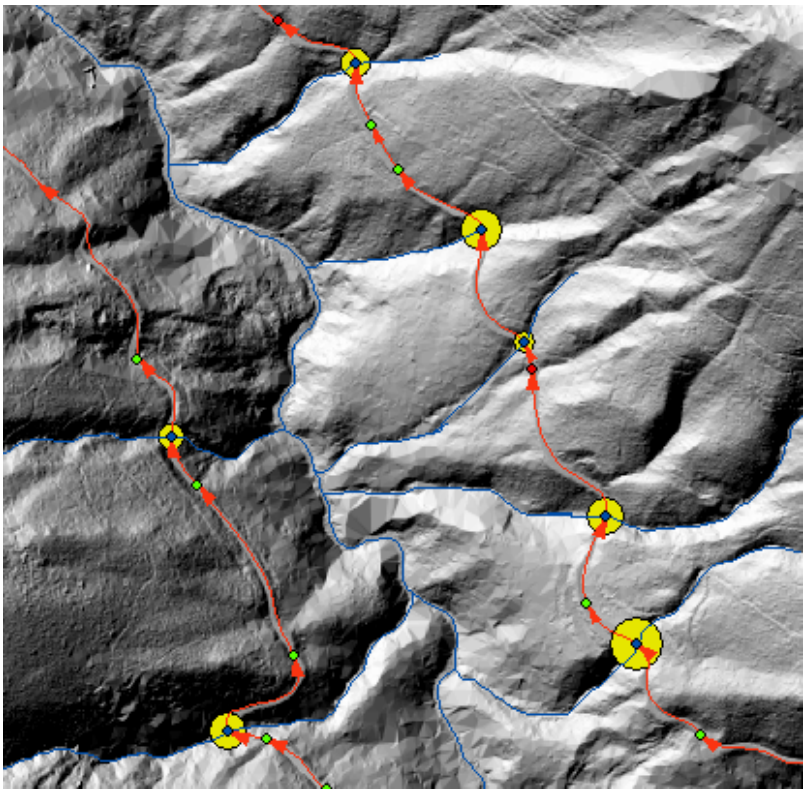


Figure 22: Topographic detail of North Tahoma, 6ft resolution DEM

As with most other computer-modeling of natural phenomena, cross drain modeling needs to be validated by field verifications. Since the local conditions at various culvert locations could be complex, input parameters to the model could be erroneous or only partially descriptive. Therefore field inspections are required in order to asses both the validity of the input factors and final outcomes at project completion. The North Tahoma Project has benefited from input validation performed by University of Washington forest engineering students, class of 2003. However, as this project was investigational and will not be implemented in its current form, no further attempts of validation have been made. A future expansion of this project might take into consideration field visits and/or a monitoring program to culvert sites in the event of a possible commission.

List of References

Burroughs E R, King J G. 1989. Reduction of soil erosion on forest roads. U.S. Department of Agriculture, Forest Service, Intermountain Research Station.

Elliot, W J, Hall D E, Graves, S R, Scheele, D L. 1999a. The X-DRAIN Cross Drain Spacing and Sediment Yield Program Version 2.00. U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, San Dimas Technology and Development Center.

Elliot, W J, Hall, D E, Scheele, D L. 1999b. Forest Service Interfaces for the Water Erosion Prediction Project Computer Model. U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, San Dimas Technology and Development Center.

Elliot W J, Hall D E, Scheele D L. 1999c. WEPP Interface for Predicting Forest Road Runoff, Erosion and Sediment Delivery. U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, San Dimas Technology and Development Center.

Ketcheson G L, Megahan W F. 1996. Sediment Production and Downslope Sediment Transport from Forest Roads in Granitic Watersheds. U.S. Department of Agriculture, Forest Service, Intermountain Research Station.

National Center for Air and Stream Improvement. 2002. Technical Documentation for SEDMODL version 2.

Reid L M, Dunne T. 1984. Sediment Production from Forest Road Surfaces. Water Resources Research 20(11).

Schiess P, Whitaker CA. 1986. Road Design and Construction in Sensitive Watersheds. Food and Agriculture Organization of the United Nations, Rome, Italy.

Washington Forest Practices Boards. 1997. Standard Methodology for Conducting Watershed Analysis Manual, version 4. WA Department of Natural Resources.

Washington Forest Practices Board. 2000. Forest Practices Board Manual, Section 3. WA Department of Natural Resources.

Watershed GeoDynamics, Megahan W F, Terra GIS Solutions. 2003. Washington Road Surface Erosion Model. WA Department of Natural Resources.

Appendix A – CULSED Software Manual

Program Requirements

CULSED has been developed and tested for ArcMap version 8.2 on the Windows 2000 platform. The application has not been tested on any subsequent versions of the ESRI software.

Installation

CULSED is comprised of two ArcMap extensions currently named RoadSedimentAnalyst and RSASedimentModel. Upon successful installation they will show up in the tools/extensions menu item of ArcMap. A dedicated toolbar provided with the program will also be found in the ArcMap list of tools.

Installation steps:

- Download the RoadSedimentAnalyst.dll and RSASedimentModel.dll to a permanent directory on your computer.
- Use the regsvr32 application generally found at \WINNT\system32\regsvr32.exe to register these dll libraries with the operating system. You can do that by dragging the libraries onto a regsvr32 icon on your desktop.
- Run the ESRI program called Component Category Manager. It can be found in \arcgis\arcexe82\Bin\categories.exe. A screen capture of this program is seen in Figure 23.
- Highlight ESRI MX Extensions. Click “Add Object” and select the RSASedimentModel.dll from the directory where you downloaded it and click “OPEN”. Select RSASedimentModel and click “OK”. This will register the RSASedimentModel extension with ArcMap.
- Similarly to step 4, register clsExt from RoadSedimentAnalyst.dll with ESRI MX Extension. Do not register any other objects here.

- Under ESRI MX CommandBars category in the Component Category Manager register clsMenu and clsToolBar from the RoadSedimentAnalyst.dll. Do not register any other objects here.
- Register all remaining objects (except clsExt, clsMenu and clsToolBar) from RoadSedimentAnalyst.dll with the ESRI MX Commands category.
- Start ArcMap and verify that the two extensions were added and the CULSED toolbar is available. If so you are ready to use the program.

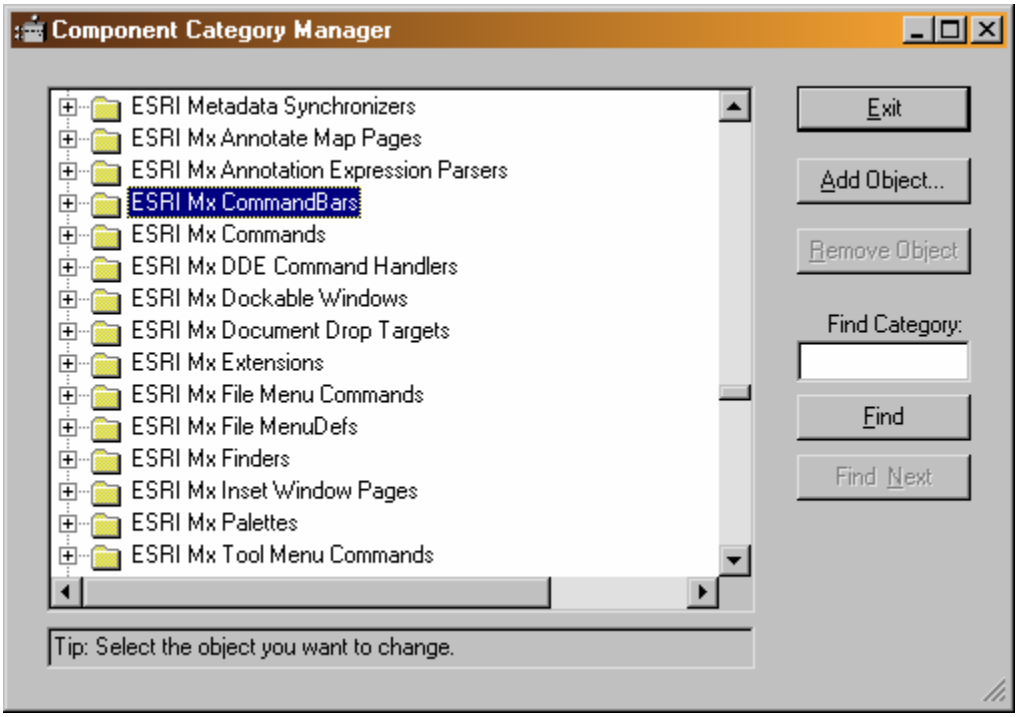


Figure 23: The Component Category Manager Application for registering CULSED components

Data Requirements

The CULSED analysis requires the following datasets:

- A hydro compensated digital elevation model. This dataset must be free of sinks and be able to derive a stream layer with the minimum contributing area method.
- A stream layer of linear features that must align with the digital elevation model. Optimally this stream layer should be generated from the DEM.

- A culvert point layer that could contain existing culvert locations. If there are no existing location the layer is still required and should be empty.
- A road layer of linear feature. The road segments should be representative of the changes in sediment producing factors (i.e. grade, width, surface material, etc.) which are given as attributes. The valid values of these attributes are given in the options menu (see below).

Software Tutorial

This section presents the tools and menu items present on the CULSED toolbar (Figure 24).

The CULSED toolbar is composed of 9 tools and 6 menu items that operate on the existing data and control the flow of the culvert analysis session. A small window on the toolbar presents the total amount of sediment delivered by the analyzed road network at each step during analysis. The sediment volume is expressed in tons / year.

All CULSED operations must be performed within an analysis session. The session steps must be performed in order. If the session is not completed in one sitting the user can leave it open in order to be stored with the ArcMap project. When the session is stopped all progress is cleared and all internal variables reinitialized.

Note: the Spatial Analyst extension must be installed for CULSED to function properly.

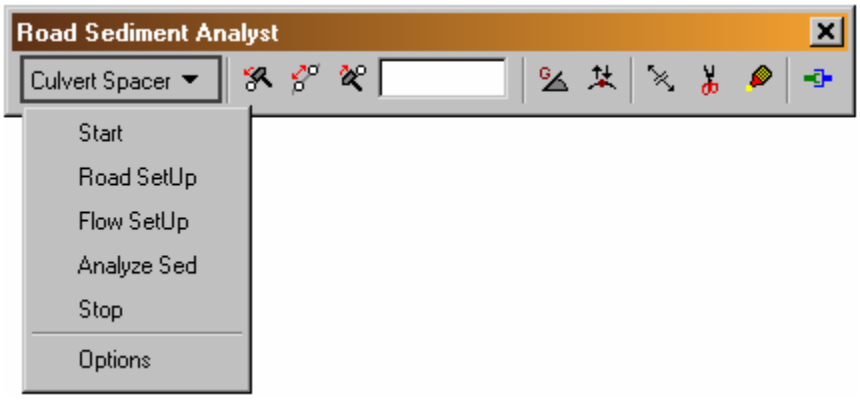


Figure 24: The CULSED toolbar

Start

This item starts an analysis session. The user is prompted to provide the needed analysis layers (Figure 25).

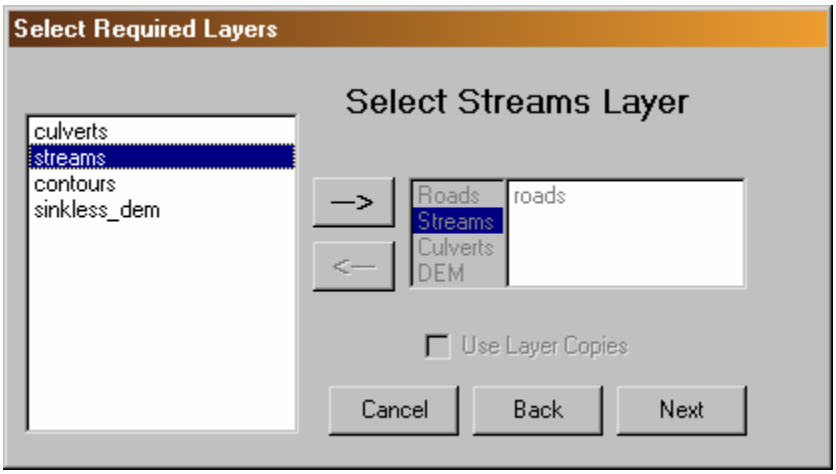


Figure 25: Start menu item

CULSED requires the following layers: a digital elevation model, a roads layer, a culvert point layer and a stream layer.

Notes: for best results the digital elevation model must be free of sinks and the stream layer must align with it. One convenient method for generating streams from a digital elevation model is the minimum contributing area method.

Road SetUp

This menu item runs a series of algorithms that setup the road network from a geometric perspective. A road grade in percent is estimate and stored in the road attribute table (Figure 26).

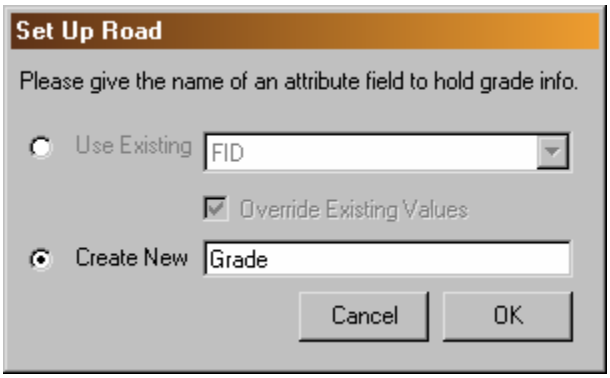


Figure 26: Set up road grade menu.

CULSED assumes that each road segment presents a consistent combination of sediment production factors (e.g. road grade, road width, surface material etc). At this stage in the analysis the user must examine the road segments and ensure that this condition is met. Because the road grade is estimated from a digital elevation model it may not necessarily reflect the actual road reality. CULSED provides tools for splitting and merging road segments, changing flow direction and modifying grade.

Note: to properly estimate the road grade the vertical units of the DEM must be the same with the horizontal units. If they differ the user must correct the grade values in the attribute table appropriately.

This tool changes the grade of a road segment. Click on a segment to display grade. Type a new value and right click to commit the change.

This tool simultaneously changes the grade of all road segments at a certain intersection. Left click to increase grade. Right click to decrease grade.

This tool changes the flow direction along a road segment. The flow direction along the side ditch is represented with arrows. Use this tool when the DEM estimated grade is incorrect.

This tool splits a road segment in two parts. All attributes are copied onto the newly created segments.

This tool merges two road segments into one. The road grade is taken from one of the segments. When more than two segments meet at an intersection click the intersection with the left button and while keeping the button depressed press the right button to select which segments will get merged.

This command enforces geometric consistency, eliminating duplicate segments and forcing node connectivity (simplifies geometries) to ensure proper topology. Make sure to use this command when you are done with all edits and are ready to proceed to next step.

Flow SetUp

This menu items runs a series of algorithms that construct the internal logical network and establish the how water flows along the side ditch. The road segments are identified as parents and children based on their physical connectivity. All existing culverts are snapped to the road lines and new culverts are placed at stream crossings.

In certain cases, the network topology algorithms cannot automatically identify the parent-child relationships needed for modeling the water travel. A list of these cases is

presented to the user during flow set-up. The user must click on each item on this list and using the ArcMap selection tool direct the water on its way to the next ditch segment (Figure 27).

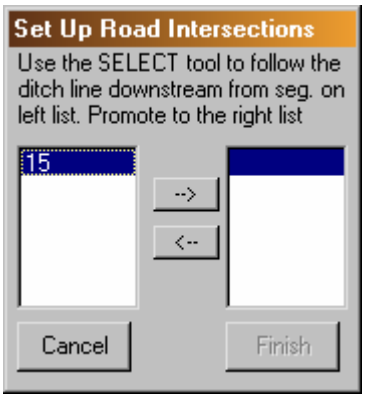


Figure 27: Set up road intersections with more than one water routing choice.

Analyze Sed

This menu item runs the sediment modeling algorithms that associate a sediment production to each road segment and determine the flow path distance from each potential culvert location to the nearest stream. This module works in conjunction with the spatial analyst extension to perform raster calculations and attribute modifications. The sediment model is implemented as a separate extension in order to provide interchangeability at run time. The options menu specifies which available sediment model is currently used.

The default sediment model provided with this version of CULSED follows the procedures in the WA DNR Manual for Conducting Watershed Analysis. The following sediment production parameters can be specified as road attributes and are associated to each segment: age, grade, width, surface material traffic and side slope cover. Other parameters such as precipitation and parent material are considered uniform over the entire study area (Figure 28). If no attributes fields are specified, a set of default road characteristics are applied. The can be viewed and modified in the option menu.

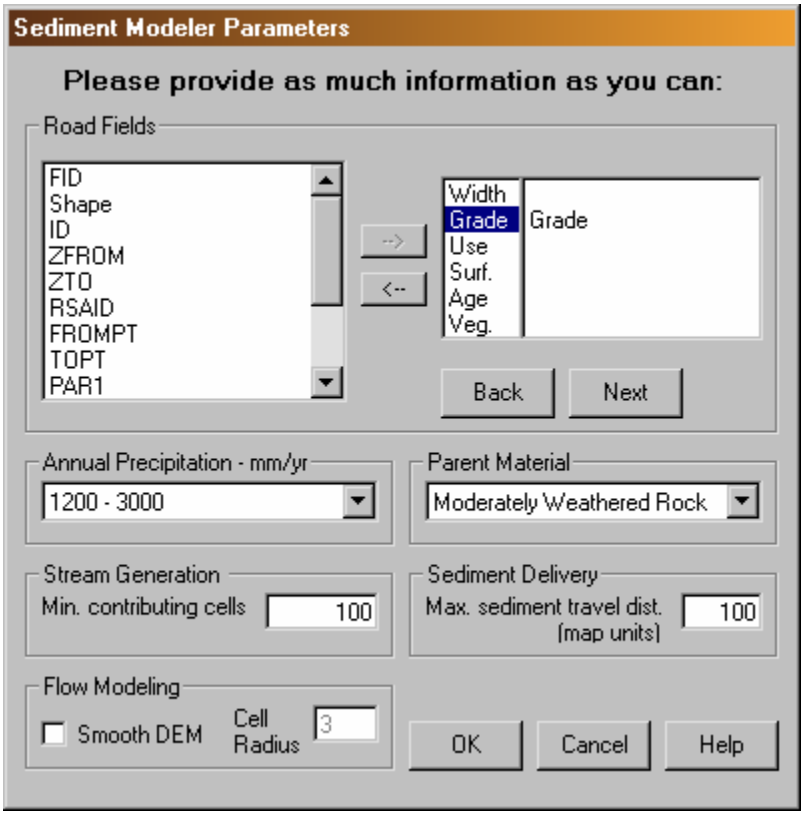


Figure 28: Sediment modeling parameters

A minimum number of contributing cells is required for computing the flow paths to the nearest streams. A raster layer of streams is generated during this process. For accurate results this number should produce a stream layer identical to the one used in the input section.

The flow modeling section is useful when working with high resolution elevation models. A typical problem when modeling water flow in these cases is the stream capture by the road ditch. To reduce the stream capturing effects the elevation models can be smoothed with a circular neighborhood of given radius.

The maximum sediment travel distance represents is a generic number that influences the sediment deposition factor used for calculating a probability of sediment delivery. This number is particular to local conditions and should be based on empirical observations at the site. The user's expertise is important for obtaining valid results.

Culvert Operation Tools

A set of tools that allow insertion, relocation and removal of cross drain culverts is provided. These tools only become available after the sediment analysis has been performed. Operating any of these tools triggers recalculations of sediment delivery probability and summation of total sediment delivered by the road network. These changes are reflected in the sediment window.

This tool inserts a new cross drain in the road drainage system.

This tool moves a cross drain culvert to a different location.

This tool removes a cross drain from the road drainage system.

Stop

Stop an analysis session and clears all variable. Use only when a session is completed.

Options

The options menu presents specifies the sediment model to be used and give the default road characteristics for sediment production calculations.

A new ArcMap extension must be developed in order to use a different sediment model. This extension must be written in Visual Basic and must implement the ISedimentModel interface provided with the CULSED code (Appendix B). This interface specifies the methods necessary for CULSED to be able to integrate with a sediment model.

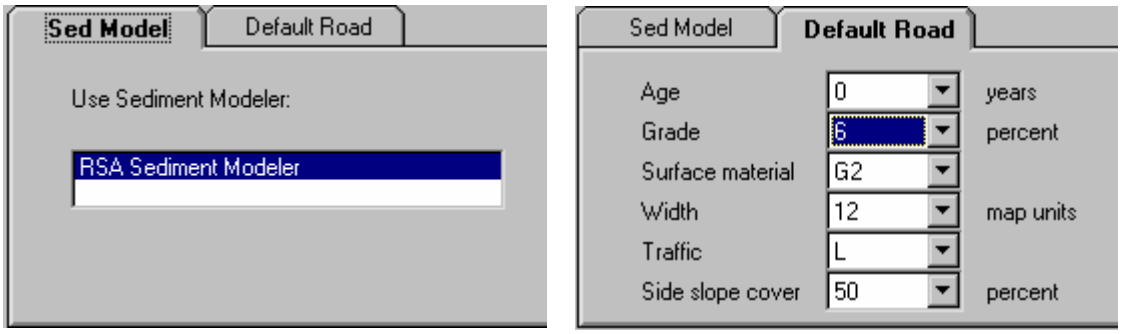


Figure 29: Option menu. Default Sediment Model (left), Default Road (right).

Workflow Example

Following the steps in the program's main menu (see above) in the order they are given guides the user through a CULSED session. The following images are screen capture of a typical CULSED session.

After inputting the required layers at the start menu the user must proceed to set up the road geometry (if needed) by estimating and adjusting grades and inspecting road attributes that drive sediment production (Figure 30). At this stage it is helpful to display the road grade associated to each segment.

The next step is to generate the road network topology. The user may be asked for sediment routing information in cases where two or more possible flow paths exist. Upon successful completion the road network is ready to be analyzed for sedimentation (Figure 31).

Special note regarding switchbacks:

CULSED does not automatically account for the “ditch-out” normally present at switch back. To prevent the program from carrying the water onto around the switchback the user must represent the ditch-out by placing a cross-drain at the center of the switchback. This cross drain is generic and can be flagged in the attribute table for later identification.

Once the sediment analysis has been run, an amount of sediment delivered will be associated to each existing culvert.

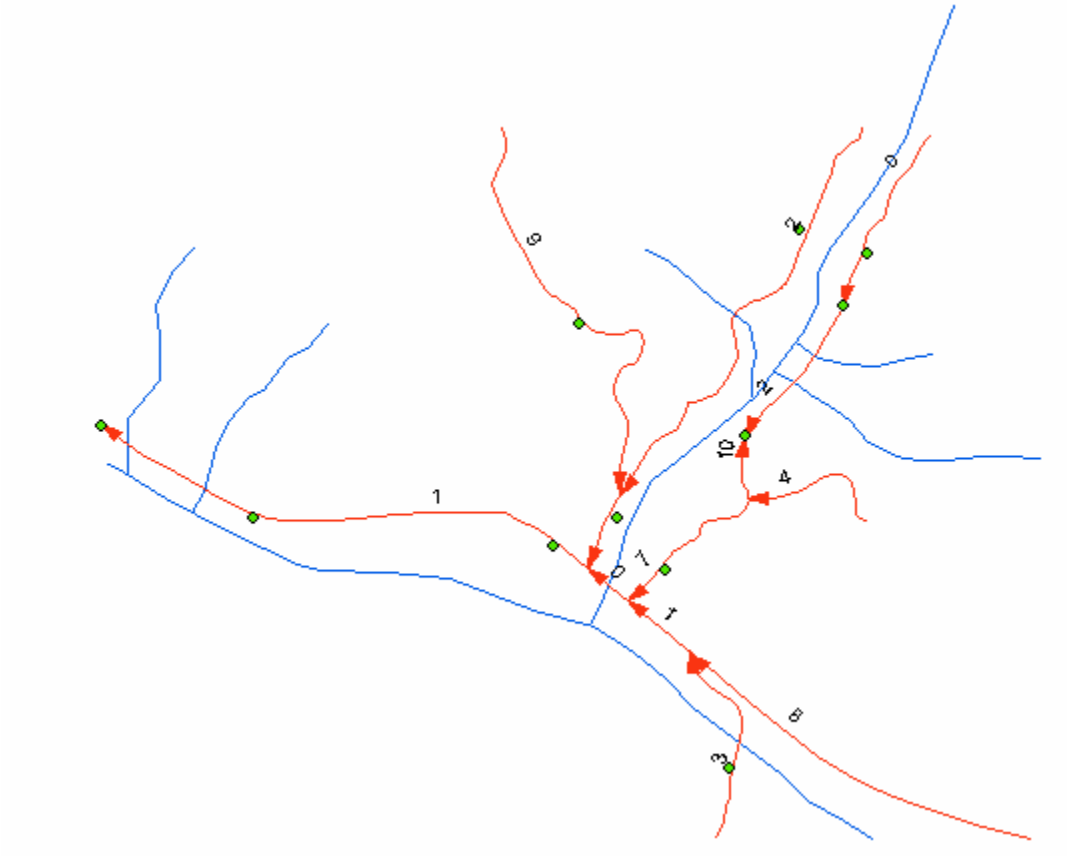


Figure 30: Road Geometry Setup has been run. A road grade is computed. The user splits segments and changes all incorrect grades appropriately.

To take advantage of the intended graphic comparison the used must draw culverts with proportional symbols based on the field called “SED” in the culvert layer’s attribute

table. If these values are spread over a wide range it is helpful to “stratify” the analysis and start by representing only the big contributors. As sedimentation is reduced by moving culverts to different locations culverts in the lower sediment ranges can also be included (Figure 32). Most of the sedimentation will be located near the stream crossings and it is these locations where users can make the greatest improvements to their drainage systems.

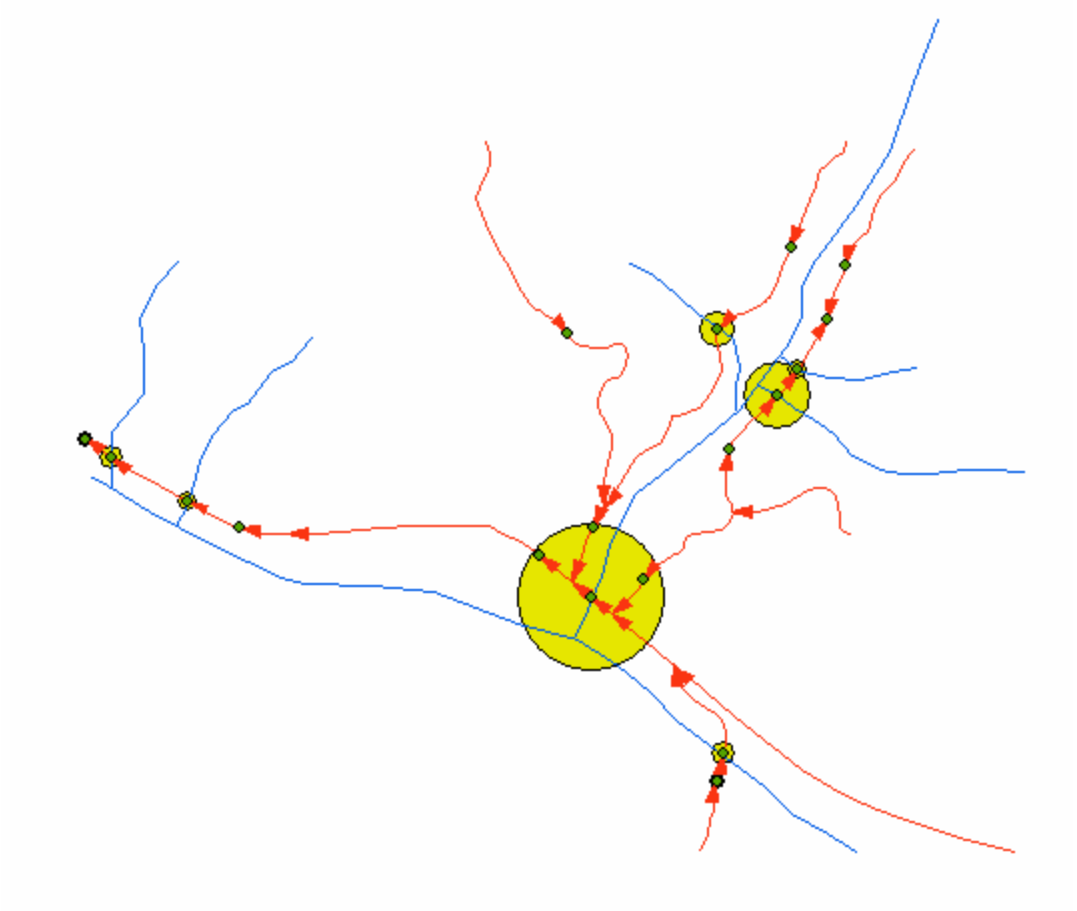


Figure 31: Flow Setup and Sediment Analysis have been run. The existing culverts are snapped to roads, culverts are automatically placed at stream crossings and a volume of sediment delivered is associated with each culvert. The user places cross drains to reduce sedimentation.

Figure 32: Culverts have been moved to near optimal locations. Sedimentation is reduced.

Appendix B – CULSED Visual Basic Computer Code

Common Implemetation Interface Code

INTERFACE - ISedimentModel (ISedimentModel.cls)

```
Option Explicit
'Standard interface ISedimentModel is meant to provide common
functionality
'across diferent implementations of sediment modelers
'The Culvert Sediment Analyst makes calls to all methods here at a
different times
'during the setup/design process

Public Function GetDeliveryPotential(pQueryPoint As IPoint) As Double
  'your implementation here
End Function

Public Function GetSedimentProduction(pRoadSegment As IFeature) As
Double
  'your implementation here
End Function

Public Function RunRasterAnalysis(pRoadClass As IFeatureClass,
pElevationModel As IRaster) As Boolean
  'your implementation here
End Function

Public Sub StopSession()
  'your implementation here
End Sub

Public Property Get DistanceToStream() As IRasterLayer

End Property

Public Property Let DistanceToStream(ByVal vNewValue As IRasterLayer)

End Property

Public Property Get MaxDeliveryDistance() As Integer

End Property

Public Property Let MaxDeliveryDistance(ByVal vNewValue As Integer)

End Property

Public Property Let DefaultRoadAge(ByVal vNewValue As Integer)

End Property

Public Property Let DefaultSlopeCover(ByVal vNewValue As Integer)

End Property
```

```
Public Property Let DefaultRoadWidth(ByVal vNewValue As Integer)

End Property

Public Property Let DefaultRoadGrade(ByVal vNewValue As Integer)

End Property

Public Property Let DefaultRoadSurface(ByVal vNewValue As String)

End Property

Public Property Let DefaultRoadTraffic(ByVal vNewValue As String)

End Property
```

Default Sediment Model Code

FORM – frmSedModelParam (frmSedModelParam.frm)

```
Private iWIndex As Integer
Private m_pFClass As IFeatureClass
Private m_bCanceled As Boolean

Private Sub cmdBack_Click()
  If iWIndex > 0 Then
    iWIndex = iWIndex - 1
    lsbWanted.ListIndex = iWIndex
    cmdNext.Enabled = True
  Else
    cmdBack.Enabled = False
  End If
  If StrComp(lsbChoices.List(iWIndex), "") Then
    Toggle cmdOut, cmdIn
  Else
    Toggle cmdIn, cmdOut
  End If
End Sub

Private Sub cmdCancel_Click()
  m_bCanceled = True
  Me.Hide
End Sub

Private Sub cmdIn_Click()
  AddToChoices
End Sub

Private Sub cmdNext_Click()
  'advance the list index by one
  If iWIndex < lsbWanted.ListCount - 1 Then
    iWIndex = iWIndex + 1
    lsbWanted.ListIndex = iWIndex
```

```
    cmdBack.Enabled = True
  Else
    cmdNext.Enabled = False
  End If
  If StrComp(lsbChoices.List(iWIndex), "") Then
    Toggle cmdOut, cmdIn
  Else
    Toggle cmdIn, cmdOut
  End If
End Sub

Private Sub cmdOK_Click()
  'validate field data
  If Not Util.ValidatePosInt(tboMinContrib.Text) Then
    MsgBox "Minimum contributing cells must be positive integer number!"
    tboMinContrib.SetFocus
    Exit Sub
  End If
  If Not Util.ValidatePosInt(tboMaxDist.Text) Then
    MsgBox "Maximum distance must be positive integer number!"
    tboMaxDist.SetFocus
    Exit Sub
  End If

  m_bCanceled = False
  'hide form
  Me.Hide
End Sub

Private Sub cmdOut_Click()
  lsbFields.AddItem lsbChoices.List(lsbWanted.ListIndex)
  'remove element from choices at index in wanted
  lsbChoices.List(lsbWanted.ListIndex) = ""
  'make in but available
  Toggle cmdIn, cmdOut
End Sub

Private Sub chkSmooth_Click()
  If chkSmooth.Value = 1 Then
    tboRadius.Enabled = True
  Else
    tboRadius.Enabled = False
  End If
End Sub

Private Sub Form_Load()

  LoadFields

  lsbWanted.AddItem "Width"
  lsbWanted.AddItem "Grade"
  lsbWanted.AddItem "Use"
  lsbWanted.AddItem "Surf."
  lsbWanted.AddItem "Age"
  lsbWanted.AddItem "Veg."

  lsbChoices.AddItem ""
  lsbChoices.AddItem ""
  lsbChoices.AddItem ""
```

```
  lsbChoices.AddItem ""
  lsbChoices.AddItem ""
  lsbChoices.AddItem ""
  'select first element on each list
  iWIndex = 0
  lsbWanted.ListIndex = 0
  'make remove btn unavailble
  cmdOut.Enabled = False
  cmdBack.Enabled = False
  chkSmooth.Value = False
  tboRadius.Enabled = False

  cboPrecip.AddItem "< 1200"
  cboPrecip.AddItem "1200 - 3000"
  cboPrecip.AddItem "> 3000"
  cboPrecip.ListIndex = 0

  cboParMat.AddItem "Mica Schist"
  cboParMat.AddItem "Volcanic Ash"
  cboParMat.AddItem "Higly Weathered Sedimentary"
  cboParMat.AddItem "Quartzite"
  cboParMat.AddItem "Course-grained Granite"
  cboParMat.AddItem "Fine-grained Granite"
  cboParMat.AddItem "Moderately Weathered Rock"
  cboParMat.AddItem "Sedimentary Rocks"
  cboParMat.AddItem "Compentent Granite"
  cboParMat.AddItem "Basalt"
  cboParMat.AddItem "Metamorphic Rocks"
  cboParMat.AddItem "Relatively Unweathered Rocks"
  cboParMat.ListIndex = 0

  tboMinContrib.Text = 100
  tboMaxDist.Text = 200
  tboRadius.Text = 3
End Sub
Private Sub Toggle(ButOn As CommandButton, ButOff As CommandButton)
  ButOn.Enabled = True
  ButOff.Enabled = False
End Sub

Private Sub Form_Unload(Cancel As Integer)
  Set m_pFClass = Nothing
End Sub

Private Sub lsbFields_DblClick()
  AddToChoices
End Sub

Private Sub lsbWanted_Click()
  lsbWanted.ListIndex = iWIndex
End Sub

Private Sub AddToChoices()
  If lsbFields.ListIndex >= 0 Then
    'put the selected element into the choices lsb
    lsbChoices.List(lsbWanted.ListIndex) =
lsbFields.List(lsbFields.ListIndex)
    'remove element from lsb fields
    lsbFields.RemoveItem (lsbFields.ListIndex)
```

```
    'make remove btn available
    Toggle cmdOut, cmdIn
  End If
End Sub

Private Sub LoadFields()
  'load fields
  If Not m_pFClass Is Nothing Then
    Dim pFields As IFields
    Set pFields = m_pFClass.Fields
    Dim i As Integer
    For i = 0 To pFields.FieldCount - 1
      lsbFields.AddItem pFields.Field(i).name
    Next i
  End If
End Sub

Public Sub InitializeData(ByVal pFeatClass As IFeatureClass)
  Set m_pFClass = pFeatClass
End Sub

Public Property Get WasCanceled() As Boolean
  WasCanceled = m_bCanceled
End Property

Public Property Get ParMatErosionCategory() As Integer
  Select Case cboParMat.List(cboParMat.ListIndex)
    Case "Mica Schist", "Volcanic Ash", "Higly Weathered Sedimentary"
      ParMatErosionCategory = 4
    Case "Quartzite", "Course-grained Granite"
      ParMatErosionCategory = 3
    Case "Fine-grained Granite", "Moderately Weathered Rock",
"Sedimentary Rocks"
      ParMatErosionCategory = 2
    Case "Compentent Granite", "Basalt", "Metamorphic Rocks",
"Relatively Unweathered Rocks"
      ParMatErosionCategory = 1
  End Select
End Property

Public Property Get AnnualPrecipCategory() As Integer
  Select Case cboPrecip.List(cboPrecip.ListIndex)
    Case "< 1200"
      AnnualPrecipCategory = 1
    Case "1200 - 3000"
      AnnualPrecipCategory = 2
    Case "> 3000"
      AnnualPrecipCategory = 3
  End Select
End Property

Public Property Get MinContrib() As Integer
  MinContrib = CInt(tboMinContrib.Text)
End Property

Public Property Get MaxDeliveryDistance() As Integer
  MaxDeliveryDistance = CInt(tboMaxDist.Text)
End Property
```

```
Public Property Get SmoothRadius() As Integer
  If tboRadius.Enabled Then
    SmoothRadius = CInt(tboRadius.Text)
  Else
    SmoothRadius = 0
  End If
End Property

Public Property Get RoadUseFieldIndex() As Long
  RoadUseFieldIndex = GetFieldIndex("Use")
End Property

Public Property Get RoadAgeFieldIndex() As Long
  RoadAgeFieldIndex = GetFieldIndex("Age")
End Property

Public Property Get RoadWidthFieldIndex() As Long
  RoadWidthFieldIndex = GetFieldIndex("Width")
End Property

Public Property Get BankVegFieldIndex() As Long
  BankVegFieldIndex = GetFieldIndex("Veg.")
End Property

Public Property Get RoadSurfFieldIndex() As Long
  RoadSurfFieldIndex = GetFieldIndex("Surf.")
End Property

Public Property Get RoadGradeFieldIndex() As Long
  RoadGradeFieldIndex = GetFieldIndex("Grade")
End Property

Public Function GetFieldIndex(sFieldTypeName As String) As Long
  If Not m_pFClass Is Nothing Then
    Dim sFieldName As String
    sFieldName = lsbChoices.List(Util.GetItemIndex(lsbWanted,
sFieldTypeName))
    If StrComp(sFieldName, "", vbBinaryCompare) <> 0 Then
      GetFieldIndex = m_pFClass.FindField(sFieldName)
      Exit Function
    End If
  End If
  GetFieldIndex = -1
End Function
```

MODULE - MeanSlopeFnct (RSAMeanSlope.bas)

```
Option Explicit

Function BufferLineFeatures(pFeatClass As IFeatureClass, distance As
Double) As IFeatureClass
  On Error GoTo erh
  'create a new feature class to hold the buffers
  '   get the feature workspace
  Dim pDataset As IDataset
  Set pDataset = pFeatClass
```

```
  '   get the spatial ref
  Dim pGeoDs As IGeoDataset
  Set pGeoDs = pFeatClass
  '   make name for new dataset
  Dim sName As String
  sName = Util.CreateUniqueRandomName(pDataset.Workspace, "buff",
esriDTFeatureClass)
  '   open a shapefile workspace to enforce making of shapefile id
featclass is coverage
  Dim pWKSfactory As IWorkspaceFactory
  Set pWKSfactory = New ShapefileWorkspaceFactory
  Dim pWks As IFeatureWorkspace
  Set pWks = pWKSfactory.OpenFromFile(pDataset.Workspace.PathName, 0)
  '   create new feature class
  Dim pNewClass As IFeatureClass
  Set pNewClass = Util.CreateWorkspaceFeatureClass(pWks, sName,
esriFTSimple, _
                                                   esriGeometryPolygon, , , ,
, pGeoDs.SpatialReference)
  'buffer features one at a time
  Dim pWKSEdit As IWorkspaceEdit
  Set pWKSEdit = pDataset.Workspace
  '   get feat cur into all polylines
  Dim pFeatCur As IFeatureCursor
  Set pFeatCur = pFeatClass.Search(Nothing, False)
  Dim pBuff As IFeature
  Dim pTopoOp As ITopologicalOperator
  Dim pLineFeat As IFeature
  Dim pPolyline As IPolyline
  Set pLineFeat = pFeatCur.NextFeature
  pWKSEdit.StartEditing False
    Do While Not pLineFeat Is Nothing
      Set pPolyline = pLineFeat.Shape
      Set pTopoOp = pPolyline

      Set pBuff = pNewClass.CreateFeature
      Set pBuff.Shape = pTopoOp.Buffer(distance)
      pBuff.Value(pBuff.Fields.FindField("Id")) = pLineFeat.OID
      pBuff.Store

      Set pLineFeat = pFeatCur.NextFeature
    Loop
  pWKSEdit.StopEditing True

  'return the new class
  Set BufferLineFeatures = pNewClass
  Exit Function
erh:
  MsgBox "error in buffering " & Error
  If Not pWKSEdit Is Nothing Then
    pWKSEdit.StopEditing False
  End If
End Function

Function ComputeZonalStat(pFeatClass As IFeatureClass, pRaster As
IRaster) As ITable
  On Error GoTo erh
  'create new ZonalOperator
  Dim pZonalOp As IZonalOp
```

```
  Set pZonalOp = New RasterZonalOp
  ' Set output workspace
  Dim pEnv As IRasterAnalysisEnvironment
  Set pEnv = pZonalOp
  Dim pDs As IDataset
  Set pDs = pFeatClass
  Set pEnv.OutWorkspace = pDs.Workspace
  'call the zonal function
  Dim pResultTable As ITable
  Set pResultTable = pZonalOp.ZonalStatisticsAsTable(pFeatClass,
pRaster, True)
  'return table
  Set ComputeZonalStat = pResultTable
  Exit Function
erh:
  MsgBox "error in RSA-ComputeZonalStat: " & Error
End Function

Sub TransferZonalSlopeToBuffers(pBuffClass As IFeatureClass, pMeanTable
As ITable)
 On Error GoTo erh
  'check to see workspace is not in editing mode
  Dim pDs As IDataset
  Set pDs = pBuffClass
  Dim pWKSEdit As IWorkspaceEdit
  Set pWKSEdit = pDs.Workspace
  If pWKSEdit.IsBeingEdited Then
    MsgBox "AddMeanField: workspace is currently being edited. exiting"
    Exit Sub
  End If
  'MsgBox "transfer1"
  'add a field to the pFeatClass
  Util.AddFieldToFeatureClass pBuffClass, "MEANSLOPE"
  'MsgBox "transfer2"
  'find the needed indexes
  Dim lMeanIdx As Long
  lMeanIdx = pMeanTable.FindField("MEAN")
  Dim lMSlopeIdx As Long
  lMSlopeIdx = pBuffClass.FindField("MEANSLOPE")
  'find the key field name
  Dim sKey As String
  sKey = pMeanTable.OIDFieldName
  'MsgBox "transfer3"
  'Start editing
  pWKSEdit.StartEditing False
  'MsgBox "transfer4"
  'Create cursor into all features
  Dim pFeatCur As IFeatureCursor
  Set pFeatCur = pBuffClass.Search(Nothing, False)
  Dim pFeat As IFeature
  Set pFeat = pFeatCur.NextFeature
  Dim pRow As IRow
  Dim pTableCur As ICursor
  Dim pTabFilter As IQueryFilter
  Set pTabFilter = New QueryFilter
  'MsgBox "transfer5!"
  Do While Not pFeat Is Nothing
    'get coresponding row
    '!!! to get the correct correspondence 1 is added to the FID
```

```
    '!!! old ArcInfo zonal stat OID's are offseted by 1
    pTabFilter.WhereClause = sKey & " = " & (pFeat.OID + 1)
    Set pTableCur = pMeanTable.Search(pTabFilter, False)
    Set pRow = pTableCur.NextRow
    If Not pRow Is Nothing Then
      pFeat.Value(lMSlopeIdx) = pRow.Value(lMeanIdx)
      pFeat.Store
    End If
    Set pFeat = pFeatCur.NextFeature
  Loop
  pWKSEdit.StopEditing True
  'MsgBox "transfer6"
 Exit Sub
erh:
  MsgBox "error in RSA-TransferZonalSlopeToBuffers: " & Err.Description
  If pWKSEdit.IsBeingEdited Then
    pWKSEdit.StopEditing False
  End If
End Sub

Function CreateSlope(pDEMRaster As Raster, pApp As IApplication) As
IRaster
  On Error GoTo erh
  ' Create a Spatial operator
  Dim pSurOp As ISurfaceOp
  Set pSurOp = New RasterSurfaceOp
  ' get dem props
  Dim pRasProps As IRasterProps
  Set pRasProps = pDEMRaster
  ' Set output environment
  Dim pEnv As IRasterAnalysisEnvironment
  Set pEnv = pSurOp
  Util.SetSpatialAnalysisSettings pEnv,
Util.GetSpatialAnalystSettings(pApp), pRasProps
  ' Perform Spatial operation
  Set CreateSlope = pSurOp.Slope(pDEMRaster,
esriGeoAnalysisSlopePercentrise)
  Exit Function
erh:
  MsgBox "error in RSA-CreateSlope: " & Err.Description
End Function

Function FillAllSinks(pDEMRaster As Raster, pApp As IApplication) As
IRaster
  On Error GoTo erh
  ' Create a Spatial operator
  Dim pHydroOp As IHydrologyOp
  Set pHydroOp = New RasterHydrologyOp
  ' get dem props
  Dim pRasProps As IRasterProps
  Set pRasProps = pDEMRaster
  ' Set output environment
  Dim pEnv As IRasterAnalysisEnvironment
  Set pEnv = pHydroOp
  Util.SetSpatialAnalysisSettings pEnv,
Util.GetSpatialAnalystSettings(pApp), pRasProps
  ' Perform Spatial operation
  Set FillAllSinks = pHydroOp.Fill(pDEMRaster)
  Exit Function
```

```
erh:
  MsgBox "error in RSA-FillSinks: " & Err.Description
End Function


Function SmoothDem(pDEMRaster As Raster, pApp As IApplication, _
                   radius As Double) As IRaster
  On Error GoTo erh
  ' Create a Spatial operator
  Dim pNeighborOp As INeighborhoodOp
  Set pNeighborOp = New RasterNeighborhoodOp
  ' get dem props
  Dim pRasProps As IRasterProps
  Set pRasProps = pDEMRaster
  ' Set output environment
  Dim pEnv As IRasterAnalysisEnvironment
  Set pEnv = pNeighborOp
  Util.SetSpatialAnalysisSettings pEnv,
Util.GetSpatialAnalystSettings(pApp), pRasProps
  ' Make a raster neighborhood
  Dim pRasNeighbor As IRasterNeighborhood
  Set pRasNeighbor = New RasterNeighborhood
  pRasNeighbor.SetCircle radius, esriUnitsCells
  ' Perform Spatial operation
  Set SmoothDem = pNeighborOp.FocalStatistics(pDEMRaster,
esriGeoAnalysisStatsMean, _
                                              pRasNeighbor, True)
  Exit Function
erh:
  MsgBox "error in RSA-SmoothDEM: " & Err.Description
End Function

Public Function GetSegmentMeanSlope(pRoadSeg As IFeature, pBuffClass As
IFeatureClass) As Double
  Dim pFilter As ISpatialFilter
  Set pFilter = New SpatialFilter
  Set pFilter.Geometry = pRoadSeg.Shape
  pFilter.SpatialRel = esriSpatialRelWithin
  Dim pFeatCur As IFeatureCursor
  Set pFeatCur = pBuffClass.Search(pFilter, False)
  Dim pBuff As IFeature
  Set pBuff = pFeatCur.NextFeature
  If Not pBuff Is Nothing Then
    GetSegmentMeanSlope =
pBuff.Value(pBuff.Fields.FindField("MEANSLOPE"))
  End If

  Set pFilter = Nothing
  Set pFeatCur = Nothing
  Set pBuff = Nothing
End Function
```

MODULE - Util (RSASedModelUtil.bas)

```
Option Explicit
```

```
Public Sub SetSpatialAnalysisSettings(pEnv1 As
IRasterAnalysisEnvironment, _
                                      pEnv2 As
IRasterAnalysisEnvironment, _
                                      Optional pRasProps As
IRasterProps)
  On Error GoTo erh
  Dim nCellSize As Double
  Dim pExtent As IEnvelope
  If Not pRasProps Is Nothing Then
    'copy Spatialreference, cellsize and extent from it
    If Not pRasProps.SpatialReference Is Nothing Then
      Set pEnv1.OutSpatialReference = pRasProps.SpatialReference
    End If
    nCellSize = pRasProps.MeanCellSize.X
    Set pExtent = pRasProps.Extent
  ElseIf Not pEnv2 Is Nothing Then
    'copy SpatiaRef, extent and cell size from passed in analysis
environment
    If Not pEnv2.OutSpatialReference Is Nothing Then
      Set pEnv1.OutSpatialReference = pEnv2.OutSpatialReference
    End If
    pEnv2.GetCellSize 3, nCellSize
    pEnv2.GetExtent 3, pExtent
  End If
  If Not pEnv2 Is Nothing Then
    'copy all the other params from the given analysis env
    Set pEnv1.OutWorkspace = pEnv2.OutWorkspace
    pEnv1.DefaultOutputRasterPrefix = pEnv2.DefaultOutputRasterPrefix
    pEnv1.DefaultOutputVectorPrefix = pEnv2.DefaultOutputVectorPrefix
    If Not pEnv2.Mask Is Nothing Then
      Set pEnv1.Mask = pEnv2.Mask
    End If
    If nCellSize <> 0 Then
      pEnv1.SetCellSize 3, nCellSize
    End If
    If Not pExtent Is Nothing Then
      pEnv1.SetExtent 3, pExtent
    End If
    pEnv1.VerifyType = pEnv2.VerifyType
  End If
  Exit Sub
erh:
    MsgBox "Failed in SetSpatialAnalysisSettings: " & Err.Description
End Sub

Public Function GetRasterDataset(pRasLayer As IRasterLayer) As
IRasterDataset
  Dim pDataset As IDataset
  Set pDataset = pRasLayer
  Dim pWks As IWorkspace
  Set pWks = pDataset.Workspace
  Dim pRasWks As IRasterWorkspace
  Set pRasWks = pWks
  Dim pRasDs As IRasterDataset
  Set pRasDs = pRasWks.OpenRasterDataset(pDataset.name)
  Set GetRasterDataset = pRasDs
  'release memory
  Set pDataset = Nothing
```

```
  Set pWks = Nothing
  Set pRasWks = Nothing
End Function

Public Sub CheckSpatialAnalystLicense()
    ' This module is used to check in the SpatialAnalyst license
    ' in a standalone VB application.
    On Error GoTo erh

    ' Get Spatial Analyst Extension UID
    Dim pUID As New UID
    pUID.Value = "esriCore.SAExtension.1"

    ' Add Spatial Analyst extension to the license manager
    Dim v As Variant
    Dim pLicAdmin As IExtensionManagerAdmin
    Set pLicAdmin = New ExtensionManager
    Call pLicAdmin.AddExtension(pUID, v)

    ' Enable the license
    Dim pLicManager As IExtensionManager
    Set pLicManager = pLicAdmin
    Dim pExtensionConfig As IExtensionConfig
    Set pExtensionConfig = pLicManager.FindExtension(pUID)
    pExtensionConfig.State = esriESEnabled
    Exit Sub
erh:
    MsgBox "Failed in License Checking" & Err.Description
End Sub

Public Function GetSpatialAnalystSettings(ByRef pApp As IApplication) As
IRasterAnalysisEnvironment
    ' This function is used to get current Spatial Analyst's Settings:
    ' RasterAnalysis Object defined through Option dialog in Spatial
    ' Analyst UI, like workspace path, cell size, extent. However, it
    ' ignores the setting of output spatial reference in the option, in
    ' other words, it is not influenced by the change of setting
    ' outout spatial reference to be the same as the data frame's.
    On Error GoTo erh
    Dim pExtension As IExtension
    Dim pSASetting As ISpatialAnalyst ' Interface for Spatial Analyst
Setting
        ' Find Spatial Analyst Extension
    Set pExtension = pApp.FindExtensionByName("Spatial Analyst")
    If Not pExtension Is Nothing Then
        ' QI IExtention for ISpatialAnalyst
        Set pSASetting = pExtension
        ' Get IRasterAnalysisEnvironment
        Set GetSpatialAnalystSettings = pSASetting.AnalysisEnvironment
    Else
        Set GetSpatialAnalystSettings = Nothing
    End If
    Set pExtension = Nothing
    Set pSASetting = Nothing
    Exit Function
erh:
    MsgBox "Failed in getting SpatialAnalyst Setting " & Err.Description
End Function
'returns a containing the cell value.
```

```
'if anything goes wrong returns nodata.
Public Function GetCellValue(pRasterLayer As IRasterLayer, pPoint As
IPoint) As String
  Dim pRIDObj As IRasterIdentifyObj
  Dim pIdentify As IIdentify
  Dim pIDArray As IArray
  Dim pNewPoint As IPoint
  Set pNewPoint = New Point
  pNewPoint.X = pPoint.X
  pNewPoint.Y = pPoint.Y
  Set pIdentify = pRasterLayer
  Set pIDArray = pIdentify.Identify(pNewPoint)
  If Not pIDArray Is Nothing Then
    Set pRIDObj = pIDArray.Element(0)
    GetCellValue = pRIDObj.name
  Else
    GetCellValue = "NoData"
  End If
  'clean up
  Set pNewPoint = Nothing
  Set pIDArray = Nothing
  Set pIdentify = Nothing
  Set pRIDObj = Nothing
End Function

Public Function ValidatePosInt(sString As String) As Boolean
  If IsNumeric(sString) And Val(sString) > 0 Then
    ValidatePosInt = True
    Exit Function
  End If
  ValidatePosInt = False
End Function

Public Function GetItemIndex(lsbList As ListBox, sItem As String) As
Integer
  Dim i As Integer
  For i = 0 To lsbList.ListCount - 1
    If StrComp(sItem, lsbList.List(i)) = 0 Then
      GetItemIndex = i
      Exit Function
    End If
  Next i
  GetItemIndex = -1
End Function

Public Function ConvertMeterTo(mapUnits As esriUnits) As Double
  Select Case mapUnits
    Case esriInches
      ConvertMeterTo = 39.37
    Case esriFeet
      ConvertMeterTo = 3.281
    Case esriYards
      ConvertMeterTo = 1.094
    Case esriMiles
      ConvertMeterTo = 0.0006212
    Case esriMillimeters
      ConvertMeterTo = 1000
    Case esriCentimeters
      ConvertMeterTo = 100
```

```
    Case esriDecimeters
      ConvertMeterTo = 10
    Case esriMeters
      ConvertMeterTo = 1
    Case esriKilometers
      ConvertMeterTo = 0.001
    Case esriNauticalMiles
      ConvertMeterTo = 0.00054
  End Select
End Function

''
'' createWorkspaceFeatureClass: simple helper to create a featureclass
in a geodatabase workspace.
'' NOTE: when creating a feature class in a workspace it is important to
assign the spatial
''       reference to the geometry field.
''
Public Function CreateWorkspaceFeatureClass(featWorkspace As
IFeatureWorkspace, _
                                            name As String, _
                                            featType As esriFeatureType,
_
                                            Optional geomType As
esriGeometryType = esriGeometryPoint, _
                                            Optional pFields As IFields,
_
                                            Optional pCLSID As UID, _
                                            Optional pCLSEXT As UID, _
                                            Optional ConfigWord As
String = "", _
                                            Optional pSpatRef As
ISpatialReference _
                                            ) As IFeatureClass

  On Error GoTo EH

  Set CreateWorkspaceFeatureClass = Nothing
  If featWorkspace Is Nothing Then Exit Function
  If name = "" Then Exit Function

  If (pCLSID Is Nothing) Or IsMissing(pCLSID) Then
    Set pCLSID = Nothing
    Set pCLSID = New UID

    '' determine the appropriate geometry type corresponding the the
feature type
    Select Case featType
      Case esriFTSimple
        pCLSID.Value = "esricore.Feature"
        If geomType = esriGeometryLine Then geomType =
esriGeometryPolyline
      Case esriFTSimpleJunction
        geomType = esriGeometryPoint
        pCLSID.Value = "esricore.SimpleJunctionFeature"
      Case esriFTComplexJunction
        pCLSID.Value = "esricore.ComplexJunctionFeature"
      Case esriFTSimpleEdge
        geomType = esriGeometryPolyline
```

```
      pCLSID.Value = "esricore.SimpleEdgeFeature"
    Case esriFTComplexEdge
      geomType = esriGeometryPolyline
      pCLSID.Value = "esricore.ComplexEdgeFeature"
    Case esriFTAnnotation
      Exit Function
  End Select
End If

' establish a fields collection
If (pFields Is Nothing) Or IsMissing(pFields) Then
  Dim pFieldsEdit As esricore.IFieldsEdit
  Set pFieldsEdit = New esricore.Fields

  ''
  '' create the geometry field
  ''
  Dim pGeomDef As IGeometryDef
  Set pGeomDef = New GeometryDef
  Dim pGeomDefEdit As IGeometryDefEdit
  Set pGeomDefEdit = pGeomDef

  ' assign the spatial reference
  Dim pSR As ISpatialReference
  If (pSpatRef Is Nothing) Or IsMissing(pSpatRef) Then
    Set pSR = New esricore.UnknownCoordinateSystem
    pSR.SetDomain 0, 21474.83645, 0, 21474.83645
    pSR.SetFalseOriginAndUnits 0, 0, 100000
  Else
    Set pSR = pSpatRef
  End If

  '' assign the geometry definiton properties.
  With pGeomDefEdit
    .GeometryType = geomType
    .GridCount = 1
    .GridSize(0) = 10
    .AvgNumPoints = 2
    .HasM = False
    .HasZ = False
    Set .SpatialReference = pSR
  End With

  Dim pField As IField
  Dim pFieldEdit As IFieldEdit
  Set pField = New Field
  Set pFieldEdit = pField

  pFieldEdit.name = "shape"
  pFieldEdit.AliasName = "geometry"
  pFieldEdit.Type = esriFieldTypeGeometry
  Set pFieldEdit.GeometryDef = pGeomDef
  pFieldsEdit.AddField pField

  ''
  '' create the object id field
  ''
  Set pField = New Field
  Set pFieldEdit = pField
```

```
    pFieldEdit.name = "OBJECTID"
    pFieldEdit.AliasName = "object identifier"
    pFieldEdit.Type = esriFieldTypeOID
    pFieldsEdit.AddField pField

    Set pFields = pFieldsEdit
  End If

  ' establish the class extension
  If (pCLSEXT Is Nothing) Or IsMissing(pCLSEXT) Then
    Set pCLSEXT = Nothing
  End If

  ' locate the shape field
  Dim strShapeFld As String
  Dim j As Integer
  For j = 0 To pFields.FieldCount - 1
    If pFields.Field(j).Type = esriFieldTypeGeometry Then
      strShapeFld = pFields.Field(j).name
    End If
  Next

  Set CreateWorkspaceFeatureClass =
featWorkspace.CreateFeatureClass(name, pFields, pCLSID,
                                 pCLSEXT, featType, strShapeFld, ConfigWord)

  Exit Function
EH:
    MsgBox Err.Description, vbInformation, "createWorkspaceFeatureClass"
End Function

Public Function CreateUniqueRandomName(pWks As IWorkspace, baseName As
String, DsType As esriDatasetType) As String
  Dim sName As String
  Math.Randomize
  Dim iRandNumber As Integer
  iRandNumber = Int(1000 * Math.Rnd + 1)
  sName = baseName & iRandNumber
  'check uniqueness end remake if necessary
  Do While Not IsUniqueName(pWks, sName, DsType)
    iRandNumber = Int(1000 * Math.Rnd + 1)
    sName = "Buff" & iRandNumber
  Loop
  CreateUniqueRandomName = sName
End Function

Public Function IsUniqueName(pWks As IWorkspace, sName As String, DsType
As esriDatasetType) As Boolean
  'return true id name is not found in this workspace
  Dim pEnumNames As IEnumDatasetName
  Set pEnumNames = pWks.DatasetNames(DsType)
  Dim pDSName As IDatasetName
  Set pDSName = pEnumNames.Next
  Do While Not pDSName Is Nothing
    If StrComp(pDSName.name, sName, vbBinaryCompare) = 0 Then
      IsUniqueName = False
      Exit Function
    End If
    Set pDSName = pEnumNames.Next
```

```
  Loop
  IsUniqueName = True
End Function

Public Sub AddFieldToFeatureClass(pFeatClass As IFeatureClass, sName As
String)
  'add a field to the pFeatClass
  Dim pFieldEdit As IFieldEdit
  Set pFieldEdit = New Field
  With pFieldEdit
    .name = sName
    .Type = esriFieldTypeDouble
  End With
  pFeatClass.AddField pFieldEdit
End Sub

Public Function GetIntegerFieldValue(pFeat As IFeature, lIndex As Long,
_
                                     iDefault As Integer) As Integer
  'see if index exists
  If lIndex > -1 Then
    'check if value is numeric
    If IsNumeric(pFeat.Value(lIndex)) Then
      GetIntegerFieldValue = pFeat.Value(lIndex)
      Exit Function
    End If
  End If
  GetIntegerFieldValue = iDefault
End Function

Public Function GetDoubleFieldValue(pFeat As IFeature, lIndex As Long, _
                                     iDefault As Double) As Double
  'see if index exists
  If lIndex > -1 Then
    'check if value is numeric
    If IsNumeric(pFeat.Value(lIndex)) Then
      GetDoubleFieldValue = pFeat.Value(lIndex)
      Exit Function
    End If
  End If
  GetDoubleFieldValue = iDefault
End Function

Public Function GetStringFieldValue(pFeat As IFeature, lIndex As Long, _
                                    iDefault As String) As String
  'see if index exists
  If lIndex > -1 Then
    GetStringFieldValue = pFeat.Value(lIndex)
    Exit Function
  End If
  GetStringFieldValue = iDefault
End Function

Public Function ConvertToAcre(units As esriUnits) As Double
  Select Case units
    Case esriInches
      ConvertToAcre = 1 / 6273000
    Case esriFeet
      ConvertToAcre = 1 / 43560
```

```
    Case esriYards
      ConvertToAcre = 1 / 4340
    Case esriMiles
      ConvertToAcre = 639.7953
    Case esriMillimeters
      ConvertToAcre = 1 / 4047000000#
    Case esriCentimeters
      ConvertToAcre = 1 / 40470000
    Case esriMeters
      ConvertToAcre = 1 / 4047
    Case esriKilometers
      ConvertToAcre = 247.0966
  End Select
End Function
```

CLASS - RSASedModel (RSASedModel.cls)

```
Option Explicit

Implements IExtension
Implements ISedimentModel
Implements IExtensionConfig

Private m_pExtState As esriExtensionState

Private m_pApp As IApplication
Private m_pDoc As IMxDocument
Private m_pDTSRasLyr As IRasterLayer
Private m_pSlopeRas As IRaster
Private m_pFilledRas As IRaster
Private m_pZonalBuffers As IFeatureClass

Private m_iPrecipCategoy As Integer
Private m_iParMatCategory As Integer
Private m_iMinContrib As Integer
Private m_iMaxDelDist As Integer
Private m_iSmoothRadius As Integer, m_iLastRadius As Integer
Private m_lRoadUseFI As Long
Private m_lRoadAgeFI As Long
Private m_lRoadWidthFI As Long
Private m_lRoadSurfFI As Long
Private m_lBankVegFI As Long
Private m_lRoadGradeFI As Long

Private m_iDefaultAge As Integer
Private m_iDefaultCover As Integer
Private m_iDefaultGrade As Integer
Private m_iDefaultWidth As Integer
Private m_sDefaultSurface As String
Private m_sDefaultTraffic As String


Private Function SurfacingCorrectionFactor(sSurface As String) As Double
  Select Case sSurface
    Case "P", "p"
      SurfacingCorrectionFactor = 0.03
```

```
    Case "DO", "do", "D-O", "d-o", "Do", "D-o"
      SurfacingCorrectionFactor = 0.15
    Case "G6", "g6", "G-6", "g-6"
      SurfacingCorrectionFactor = 0.2
    Case "G2", "g2", "G-2", "g-2"
      SurfacingCorrectionFactor = 0.5
    Case "N", "R", "n", "r"
      SurfacingCorrectionFactor = 1#
    Case Else
      SurfacingCorrectionFactor = 1#
  End Select
End Function

Private Function TrafficCorrectionFactor(sTraffic As String) As Double
  Select Case sTraffic
    'heavy traffic / active mainline
    Case "h", "H", "AM", "am", "Am"
      If m_iPrecipCategoy = 1 Then ' < 1200 mm/year
        TrafficCorrectionFactor = 20
      ElseIf m_iPrecipCategoy = 2 Then
        TrafficCorrectionFactor = 50
      Else
        TrafficCorrectionFactor = 120
      End If
    'moderate traffic / active secondary
    Case "m", "M", "AS", "as", "As"
      If m_iPrecipCategoy = 1 Then ' < 1200 mm/year
        TrafficCorrectionFactor = 2
      ElseIf m_iPrecipCategoy = 2 Then
        TrafficCorrectionFactor = 4
      Else                          ' > 3000 mm/year
        TrafficCorrectionFactor = 10
      End If
    'light traffic / not active
    Case "l", "L", "NA", "na", "Na"
      TrafficCorrectionFactor = 1
    'no traffic / abandoned
    Case "n", "N", "a", "A"
      If m_iPrecipCategoy = 1 Then
        TrafficCorrectionFactor = 0.02
      ElseIf m_iPrecipCategoy = 2 Then
        TrafficCorrectionFactor = 0.05
      Else
        TrafficCorrectionFactor = 0.1
      End If
    'everything else
    Case Else ' assume low values
      TrafficCorrectionFactor = 1
    End Select
End Function

Private Property Get IExtension_Name() As String
  IExtension_Name = "RSA Sediment Modeler"
End Property

Private Sub IExtension_Shutdown()
  Set m_pApp = Nothing
  Set m_pDoc = Nothing
  Set m_pDTSRasLyr = Nothing
```

```
End Sub

Private Sub IExtension_Startup(initializationData As Variant)
  Set m_pApp = initializationData
  Set m_pDoc = m_pApp.Document

End Sub

Private Property Get IExtensionConfig_Description() As String
  IExtensionConfig_Description = "Sediment Modeler for Road Sediment
Analyst"
End Property

Private Property Get IExtensionConfig_ProductName() As String
  IExtensionConfig_ProductName = "RSA Sediment Modeler"
End Property

Private Property Let IExtensionConfig_State(ByVal RHS As
esricore.esriExtensionState)
  m_pExtState = RHS
End Property

Private Property Get IExtensionConfig_State() As
esricore.esriExtensionState
  IExtensionConfig_State = m_pExtState
End Property

Private Function ComputeDistanceToStreams(pElevationRaster As
esricore.IRaster, _
                                          Optional lContribCells As
Long) As Boolean
  On Error GoTo erh

  'use either the value passed in or the one set by the user
  If lContribCells = 0 Then lContribCells = m_iMinContrib

  'define raster model
  Dim pRModel As IRasterModel
  Set pRModel = New RasterModel

  ' Create spatial analysis environment
  Dim pEnv As IRasterAnalysisEnvironment
  Set pEnv = pRModel

  ' Set Raster Analysis parameters
  Dim pRasProps As IRasterProps
  Set pRasProps = pElevationRaster
  Util.SetSpatialAnalysisSettings pEnv,
Util.GetSpatialAnalystSettings(m_pApp), pRasProps

  ' Set model, vbLf is used to separate equations
  Dim sScript As String
  Dim pDEMRaster As esricore.IRaster

  If m_iSmoothRadius > 0 Then 'user wants smoothing
    If m_iSmoothRadius <> m_iLastRadius Then 'must calculate rasters
      Dim pSmoothRaster As esricore.IRaster
      Dim pFilledRaster As esricore.IRaster
```

```
      m_iLastRadius = m_iSmoothRadius 'reset last radius for future
reference

      Set pSmoothRaster = MeanSlopeFnct.SmoothDem(pElevationRaster,
m_pApp, _
                                                  CDbl(m_iSmoothRadius))
      Set m_pFilledRas = MeanSlopeFnct.FillAllSinks(pSmoothRaster,
m_pApp)
    End If
    'use the raster from last pass
    Set pDEMRaster = m_pFilledRas
  Else
    Set pDEMRaster = pElevationRaster
  End If

  pRModel.Script = "[fdir] = flowdirection([dem], #, force)" & vbLf & _
                   "[facc] = flowaccumulation([fdir])" & vbLf & _
                   "[istr] = con([facc] > " & lContribCells & ", 0, 1)"
& vbLf & _
                   "[wght] = [istr] * sqrt(1 + pow([slp] / 100, 2))" &
vbLf & _
                   "[dtst] = flowlength([fdir], [wght], downstream)"

  ' Bind to raster
  pRModel.BindRaster pDEMRaster, "dem"
  pRModel.BindRaster m_pSlopeRas, "slp"

  ' Run the model
  pRModel.Execute

  ' Unbind raster
  pRModel.UnbindSymbol "dem"
  pRModel.UnbindSymbol "slp"

  ' Get outputs
  Dim pOutRas As IRaster
  Set pOutRas = pRModel.BoundRaster("dtst")
  ' Set pointer to new output raster
  Set m_pDTSRasLyr = New RasterLayer
  m_pDTSRasLyr.CreateFromRaster pOutRas
  'add layer to map
  m_pDTSRasLyr.Visible = False
  m_pDoc.AddLayer m_pDTSRasLyr
  m_pDoc.FocusMap.MoveLayer m_pDTSRasLyr, m_pDoc.FocusMap.LayerCount - 1

'  ' Make this dataset permanent
'  Dim pTempRas As ITemporaryDataset
'  Set pTempRas = Util.GetRasterDataset(m_pDTSRasLyr)
'  If pTempRas.IsTemporary Then pTempRas.MakePermanent

  ComputeDistanceToStreams = True

  'release memory
  Set pRModel = Nothing
  Set pRasProps = Nothing
  Set pEnv = Nothing
  Exit Function
erh:
  MsgBox "error in ComputeDistToStream " & Error
```

```
  ComputeDistanceToStreams = False
End Function

Public Property Let ISedimentModel_DistanceToStream(ByVal RHS As
esricore.IRasterLayer)
  Set m_pDTSRasLyr = RHS
End Property

Public Property Get ISedimentModel_DistanceToStream() As
esricore.IRasterLayer
  Set ISedimentModel_DistanceToStream = m_pDTSRasLyr
End Property

Public Function ISedimentModel_GetDeliveryPotential(pQueryPoint As
esricore.IPoint) As Double
  Dim sValue As String
  sValue = Util.GetCellValue(m_pDTSRasLyr, pQueryPoint)
  If StrComp(sValue, "NoData", vbTextCompare) <> 0 Then
    ISedimentModel_GetDeliveryPotential = CalcDelPot(CDbl(sValue))
  Else
    ISedimentModel_GetDeliveryPotential = 0#
  End If
End Function

Public Function ISedimentModel_GetSedimentProduction(pRoadSegment As
esricore.IFeature) As Double
  On Error GoTo erh

  Dim dSedProd As Double
  'get road age
  Dim iRoadAge As Integer
  iRoadAge = Util.GetIntegerFieldValue(pRoadSegment, m_lRoadAgeFI,
m_iDefaultAge)
  'basic erosion rate in tons/year/acre of road prism
  Dim iBaseSed As Integer
  iBaseSed = BasicErosionRate(iRoadAge)
  'cover factor
  Dim iCover As Integer
  iCover = Util.GetIntegerFieldValue(pRoadSegment, m_lBankVegFI,
m_iDefaultCover)
  Dim dCovFact As Double
  dCovFact = CoverCorrectionFactor(iCover, iRoadAge)
  'surfacing factor
  Dim sSurf As String
  sSurf = Util.GetStringFieldValue(pRoadSegment, m_lRoadSurfFI,
m_sDefaultSurface)
  Dim dSurfFact As Double
  dSurfFact = SurfacingCorrectionFactor(sSurf)
  'traffic factor
  Dim sTraf As String
  sTraf = Util.GetStringFieldValue(pRoadSegment, m_lRoadUseFI,
m_sDefaultTraffic)
  Dim dTrafFact As Double
  dTrafFact = TrafficCorrectionFactor(sTraf)
  'grade factor
  Dim iRoadGrade As Integer
  iRoadGrade = Math.Abs(Util.GetIntegerFieldValue(pRoadSegment,
m_lRoadGradeFI, m_iDefaultGrade))
  Dim dGradeFact As Double
```

```
  dGradeFact = GradeCorrectionFactor(iRoadGrade)

  'compute sediment production in tons/year/acre of road prism
  Dim dThreadSed As Double, dCutSlopeSed As Double
  dThreadSed = 0.4 * iBaseSed * dGradeFact * dTrafFact * dSurfFact
  dCutSlopeSed = 0.4 * iBaseSed * dCovFact

  'get surface of both the thread and cut bank
  Dim pPolyline As IPolyline
  Set pPolyline = pRoadSegment.Shape
  Dim dLength As Double, dThreadWidth As Double, dCutWidth As Double
  dLength = pPolyline.Length
  dThreadWidth = Util.GetDoubleFieldValue(pRoadSegment, m_lRoadWidthFI,
CDbl(m_iDefaultWidth))
  dCutWidth = CutSlopeWidth(pRoadSegment, dThreadWidth)
  Dim dThreadArea As Double, dCutArea As Double
  'the sediment rate is in Tons/acre/year
  'convert square map units to acre
  dThreadArea = dLength * dThreadWidth *
Util.ConvertToAcre(m_pDoc.FocusMap.mapUnits)
  dCutArea = dLength * dCutWidth *
Util.ConvertToAcre(m_pDoc.FocusMap.mapUnits)

  'return result
  ISedimentModel_GetSedimentProduction = dThreadSed * dThreadArea +
dCutSlopeSed * dCutArea

  Exit Function
erh:
  MsgBox "RSA SedModel -- error in GetSedimentProduction" & vbLf & Error
End Function
Private Function CalcDelPot(dDistToStream As Double) As Double
  'based on Ketcheson and Megahan "Sediment Production and Downslope..."
  'modification : do not allow values under 0.0001 for proportional
symbol
  'display restrictions.
  'also modified to accept different values for maxDelDistance - needs
theoretical profing
  On Error GoTo erh
  Dim dPotential As Double
  dPotential = 1.0362 * Exp(-100 * dDistToStream / (32.88 *
m_iMaxDelDist)) - 0.0555
  If dPotential < 0.001 Then
    CalcDelPot = 0
    Exit Function
  End If
  CalcDelPot = dPotential

  Exit Function
erh:
  CalcDelPot = 0.001
End Function

Private Function ComputeMeanTerrainSlope(pRoads As IFeatureClass, pElev
As IRaster) As Boolean
  On Error GoTo erh

  If m_pZonalBuffers Is Nothing Then
    'create a slope raster layer
```

```
    'MsgBox "debug 1"
    Set m_pSlopeRas = MeanSlopeFnct.CreateSlope(pElev, m_pApp)

    'buffer the roads
    'MsgBox "debug 2"
    Dim dDist As Double
    dDist = 30 * Util.ConvertMeterTo(m_pDoc.FocusMap.mapUnits)
    Dim pBuffClass As IFeatureClass
    Set pBuffClass = MeanSlopeFnct.BufferLineFeatures(pRoads, dDist)

    'MsgBox "debug 3"
    'run zonal operation on slopes to roads
    Dim pZoneTable As ITable
    Set pZoneTable = MeanSlopeFnct.ComputeZonalStat(pBuffClass,
m_pSlopeRas)
    'MsgBox "debug 4"
    'transfer mean slopes onto the buffers
    MeanSlopeFnct.TransferZonalSlopeToBuffers pBuffClass, pZoneTable
    'MsgBox "debug 5"
    'delete the leftover datasets
    Dim pDs As IDataset
    Set pDs = pZoneTable
    pDs.Delete
    'MsgBox "debug 6"

    'return true
    Set m_pZonalBuffers = pBuffClass
  End If

  ComputeMeanTerrainSlope = True

  Exit Function
erh:
  MsgBox "RSA SedModel -- error in ComputeMeanTerrainSlope" & vbLf &
Error
  ComputeMeanTerrainSlope = False
End Function

' returns the basic sediment production in tones /year /acre of road
Private Function BasicErosionRate(iAge As Integer) As Integer
  Select Case m_iParMatCategory
    Case 1
      If iAge > 2 Then
        BasicErosionRate = 10
      Else
        BasicErosionRate = 20
      End If
    Case 2
      If iAge > 2 Then
        BasicErosionRate = 30
      Else
        BasicErosionRate = 60
      End If
    Case 3
      If iAge > 2 Then
        BasicErosionRate = 30
      Else
        BasicErosionRate = 110
      End If
```

```
    Case 4
      If iAge > 2 Then
        BasicErosionRate = 60
      Else
        BasicErosionRate = 110
      End If
  End Select
End Function

' returns the cover correction factor as ratio
' pass in a negative dCover to approximate cover by age
Private Function CoverCorrectionFactor(iCover As Integer, iRoadAge As
Integer) As Double
  Dim CovValue As Integer
  CovValue = 0 ' default value, also assumed for new road
  If iCover < 0 Then
    If iRoadAge > 2 Then CovValue = 50
  Else:
    CovValue = iCover
  End If
  Select Case CovValue + 0.0001
    Case 0 To 10
      CoverCorrectionFactor = 1#
    Case 10 To 20
      CoverCorrectionFactor = 0.77
    Case 20 To 30
      CoverCorrectionFactor = 0.63
    Case 30 To 50
      CoverCorrectionFactor = 0.53
    Case 50 To 80
      CoverCorrectionFactor = 0.37
    Case Is > 80
      CoverCorrectionFactor = 0.18
  End Select
End Function
' grade is in percent rise/length
Private Function GradeCorrectionFactor(iGrade As Integer) As Double
  'CH Luce and TA Black 1999 - Sediment production from forest roads in
Oregon
  'shown erosion proportional with square of road grade
  'GradeCorrectionFactor = iGrade * iGrade / 36
  GradeCorrectionFactor = iGrade / 6
End Function

Private Function CutSlopeWidth(pRoadFeature As IFeature, dWidth As
Double) As Double
  Const PI = 3.14159265358979

  ' get the angles
  Dim alpha As Double, beta As Double
  beta = 0.785398 ' 1:1 cutslope angle in radians
  Dim dMeanSideSlope As Double
  dMeanSideSlope = MeanSlopeFnct.GetSegmentMeanSlope(pRoadFeature,
m_pZonalBuffers)
  alpha = Math.Atn(dMeanSideSlope / 100)

  ' enforce Beta to be bigger than Alpha
  If beta - alpha <= 0 Then
    beta = 1.107148 ' 2:1 cut slope in radians
```

```
    If beta - alpha <= 0 Then
      beta = alpha + PI / 180 ' add one degree to alpha
    End If
  End If

  ' determine the benched width of road
  Dim bench As Double
  Select Case (alpha * 180 / PI)
    Case Is > 55
      bench = dWidth
    Case 25 To 55
      bench = 2 * dWidth / 3
    Case Is < 25
      bench = dWidth / 2
  End Select

  'compute the cutslope length
  CutSlopeWidth = bench * Math.Sin(alpha) / Math.Sin(beta - alpha)

  Exit Function
erh:
  MsgBox "RSA SedModel -- error in CutSlopeWidth" & vbLf & Error
End Function

Public Property Let ISedimentModel_MaxDeliveryDistance(ByVal RHS As
Integer)
  m_iMaxDelDist = RHS
End Property

Public Property Get ISedimentModel_MaxDeliveryDistance() As Integer
  ISedimentModel_MaxDeliveryDistance = m_iMaxDelDist
End Property

Public Function ISedimentModel_RunRasterAnalysis(pRoadClass As
esricore.IFeatureClass, _
                                                 pElevationModel As
esricore.IRaster) As Boolean
  On Error GoTo erh

   'check license
  Util.CheckSpatialAnalystLicense

  ' Change cursor while calculating
  Dim pCur As IMouseCursor
  Dim bSuccesful As Boolean
  Dim pUserDialog As New frmSedModelParam
  pUserDialog.InitializeData pRoadClass
  pUserDialog.Show vbModal
  'dialog is modal so execution thread would interupt
  'check if dialog completed normally and was not cancel
  If Not pUserDialog.WasCanceled Then

    'get the user settings
    m_lRoadAgeFI = pUserDialog.RoadAgeFieldIndex
    m_lRoadUseFI = pUserDialog.RoadUseFieldIndex
    m_lRoadWidthFI = pUserDialog.RoadWidthFieldIndex
    m_lRoadGradeFI = pUserDialog.RoadGradeFieldIndex
    m_lBankVegFI = pUserDialog.BankVegFieldIndex
    m_lRoadSurfFI = pUserDialog.RoadSurfFieldIndex
```

```
    m_iMaxDelDist = pUserDialog.MaxDeliveryDistance
    m_iMinContrib = pUserDialog.MinContrib
    m_iSmoothRadius = pUserDialog.SmoothRadius
    m_iParMatCategory = pUserDialog.ParMatErosionCategory
    m_iPrecipCategoy = pUserDialog.AnnualPrecipCategory

    'prepare the zonal slope buffers for later use
    Set pCur = New MouseCursor
    pCur.SetCursor 2
    bSuccesful = ComputeMeanTerrainSlope(pRoadClass, pElevationModel)
    If bSuccesful Then bSuccesful =
ComputeDistanceToStreams(pElevationModel)
  Else
    bSuccesful = False
  End If

  If Not pCur Is Nothing Then pCur.SetCursor 0
  'unload form
  Unload pUserDialog
  'release memory
  Set pUserDialog = Nothing
  Set pCur = Nothing

 ISedimentModel_RunRasterAnalysis = bSuccesful
 Exit Function
erh:
  If Not pCur Is Nothing Then
    pCur.SetCursor 0
    Set pCur = Nothing
  End If
  MsgBox "RSA SedModel -- error in ComputeSedimentProduction" & vbLf &
Error
  ISedimentModel_RunRasterAnalysis = False
End Function

Public Property Let ISedimentmodel_DefaultRoadAge(ByVal vNewValue As
Integer)
  m_iDefaultAge = vNewValue
End Property

Public Property Let ISedimentModel_DefaultSlopeCover(ByVal vNewValue As
Integer)
  m_iDefaultCover = vNewValue
End Property

Public Property Let ISedimentmodel_DefaultRoadWidth(ByVal vNewValue As
Integer)
  m_iDefaultWidth = vNewValue
End Property

Public Property Let ISedimentmodel_DefaultRoadGrade(ByVal vNewValue As
Integer)
  m_iDefaultGrade = vNewValue
End Property

Public Property Let iSedimentmodel_DefaultRoadSurface(ByVal vNewValue As
String)
  m_sDefaultSurface = vNewValue
End Property
```

```
Public Property Let Isedimentmodel_DefaultRoadTraffic(ByVal vNewValue As
String)
  m_sDefaultTraffic = vNewValue
End Property

Public Sub ISedimentModel_StopSession()
  Set m_pZonalBuffers = Nothing
  Set m_pSlopeRas = Nothing
  Set m_pFilledRas = Nothing
  m_iLastRadius = 0
End Sub
```

CULSED Main Program Code

FORM – frmChildDec (frmChildDec.frm)

```
Option Explicit

Public Event HasFinished(bCancel As Boolean)

Private colDecisions As Collection
Private pLayer As IFeatureLayer
Private pFeatSel As IFeatureSelection
Private pFilter As IQueryFilter
Private pFeatCur As IFeatureCursor
Private pFeat As IFeature
Private pSelSet As ISelectionSet
Private pActiveView As IActiveView
Private lValue As Long

'the following are for showing this window always on top
Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long,
_
ByVal hWndInsertAfter As Long, ByVal X As Long, _
ByVal Y As Long, ByVal CX As Long, ByVal CY As Long, _
ByVal wFlags As Long) As Long
Private Const SWP_NOMOVE = 2
Private Const SWP_NOSIZE = 1
Private Const HWND_TOPMOST = -1
Private Const HWND_NOTOPMOST = -2

Private Sub LoadUndecided()
  'load form's list1 with the values passed from an array
  Dim colChildren As Collection
  For Each colChildren In colDecisions
    lsbList1.AddItem (colChildren.Item(1))
    lsbList2.AddItem ("")
  Next
End Sub

Private Sub btnIn_Click()
  'get selection and put first item in list2
  Set pSelSet = pFeatSel.SelectionSet
  pSelSet.Search Nothing, False, pFeatCur
  Set pFeat = pFeatCur.NextFeature
```

```
  If (Not pFeat Is Nothing) And lsbList1.ListIndex > -1 Then
    lValue = pFeat.Value(pFeat.Fields.FindField("RSAID"))
    If Exists(lsbList1.ListIndex, lValue) Then
      lsbList2.List(lsbList1.ListIndex) = lValue
    End If
  End If
  'validate list2 in order to release the finish button
  If ValidateAllEntries Then btnFinish.Enabled = True
End Sub

Private Sub btnCancel_Click()
  'hide form and destroy form data
  Me.Hide
  pFeatSel.Clear
  pFeatSel.SelectionChanged
  pActiveView.PartialRefresh esriViewGeography, pLayer, Nothing
  RaiseEvent HasFinished(True)
End Sub

Private Sub btnFinish_Click()
  'call the rest of the set-up methods, pass back an array of children
  Me.Hide
  RemoveFromCollection
  RaiseEvent HasFinished(False)
End Sub

Private Sub btnOut_Click()
  'get selected item from list1 and eliminate corespondent from list2
  If lsbList1.ListIndex > -1 Then
    lsbList2.List(lsbList1.ListIndex) = ""
  End If
  'disable the finish button
  btnFinish.Enabled = False
End Sub

Private Sub Form_Load()
  SetOnTop Me.hwnd, True
End Sub

Private Sub lsbList1_Click()
  'select correspondent item in list 2
  lsbList2.Selected(lsbList1.ListIndex) = True
  'select the feature the user clicked on
  pFilter.WhereClause = "RSAID = " & lsbList1.List(lsbList1.ListIndex)
  pFeatSel.SelectFeatures pFilter, esriSelectionResultNew, False
  pFeatSel.SelectionChanged
  pActiveView.PartialRefresh esriViewGeography, pLayer, Nothing
End Sub

'acts as a constructor. Change when implementing in VB!
Public Sub SetUpDialog(ByRef colUserDec As Collection, pFeatLayer As
IFeatureLayer, _
                       ByRef pMap As IMap)

  'pointer to the collection passed by ref
  Set colDecisions = colUserDec
  Set pLayer = pFeatLayer
  LoadUndecided
  'set up selection set
```

```
  Set pFeatSel = pLayer
  Set pFilter = New QueryFilter
  Set pSelSet = pFeatSel.SelectionSet
  Set pActiveView = pMap
End Sub
'verifies that all values have been set to valid numbers
Private Function ValidateAllEntries() As Boolean
  Dim counter As Integer
  For counter = 0 To lsbList1.ListCount - 1
    If "" = lsbList2.List(counter) Then
      ValidateAllEntries = False
      Exit Function
    End If
  Next counter
  ValidateAllEntries = True
End Function
'for each road segment remove the child the user selected
Private Sub RemoveFromCollection()
  Dim colChildren As Collection
  Dim iCounter As Integer, iListIndex As Integer
  iListIndex = 0
  For Each colChildren In colDecisions
    For iCounter = colChildren.count To 2 Step -1
      If colChildren(iCounter) = lsbList2.List(iListIndex) Then
        colChildren.Remove (iCounter)
        iCounter = iCounter - 1
      End If
    Next iCounter
    iListIndex = iListIndex + 1
  Next
  Set colChildren = Nothing
End Sub

Private Function Exists(iIndex As Integer, lValue As Long) As Boolean
  Dim colChildren As Collection
  Set colChildren = colDecisions(iIndex + 1)
  Dim i As Integer
  For i = 2 To colChildren.count
    If lValue = colChildren(i) Then
      Exists = True
      Set colChildren = Nothing
      Exit Function
    End If
  Next i
  Exists = False
  Set colChildren = Nothing
End Function

Public Sub SetOnTop(ByVal hwnd As Long, ByVal bSetOnTop As Boolean)
    Dim lR As Long
    If bSetOnTop Then
      lR = SetWindowPos(hwnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOSIZE)
    Else
      lR = SetWindowPos(hwnd, HWND_NOTOPMOST, 0, 0, 0, 0, SWP_NOSIZE)
    End If
End Sub
```

FORM - frmGradeField (frmGradeField.frm)

```
Option Explicit

'the following are for showing this window always on top
Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long,
_
ByVal hWndInsertAfter As Long, ByVal X As Long, _
ByVal Y As Long, ByVal CX As Long, ByVal CY As Long, _
ByVal wFlags As Long) As Long
Private Const SWP_NOMOVE = 2
Private Const SWP_NOSIZE = 1
Private Const HWND_TOPMOST = -1
Private Const HWND_NOTOPMOST = -2

Private Sub Form_Load()
  SetOnTop Me.hwnd, True
End Sub

Public Sub SetOnTop(ByVal hwnd As Long, ByVal bSetOnTop As Boolean)
    Dim lR As Long
    If bSetOnTop Then
      lR = SetWindowPos(hwnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOSIZE)
    Else
      lR = SetWindowPos(hwnd, HWND_NOTOPMOST, 0, 0, 0, 0, SWP_NOSIZE)
    End If
End Sub

Public Property Get GradeValue() As String
  GradeValue = tboGradeField.Text
End Property

Public Property Let GradeValue(ByVal sNewValue As String)
  tboGradeField.Text = sNewValue
  tboGradeField.SelLength = 4
End Property
```

FORM - frmGradeName.frm

```
Option Explicit

Private m_pExt As clsExt
Private m_bCompletedOk As Boolean

Private Sub cmdCancel_Click()
  m_bCompletedOk = False
  Me.Visible = False
End Sub

Private Sub cmdOK_Click()
  If Not m_pExt Is Nothing Then
    Dim sFieldName As String
    If optExists.Value Then
      'get field name
      sFieldName = cboFields.List(cboFields.ListIndex)
```

```
      'set RSA grade name
      m_pExt.GradeName = sFieldName
      'signal success
      m_bCompletedOk = True

    ElseIf optCreate.Value Then
      'get field name
      sFieldName = tboName.Text

      'verify if field exsits
      If Not Util.ExistsField(m_pExt.RoadLayer.FeatureClass, sFieldName)
Then
        'show dialog
        Dim response As VbMsgBoxResult
        response = MsgBox("Field " & sFieldName & " already exists!" &
vbLf & "Use it?", vbYesNo)
        If response = vbNo Then Exit Sub ' return to the form
      End If

      'create field
      If Util.AddField(m_pExt.RoadLayer.FeatureClass, sFieldName,
esriFieldTypeInteger) Then
        'set RSA grade name
        m_pExt.GradeName = sFieldName
        'signal success
        m_bCompletedOk = True
      Else
        m_bCompletedOk = False
      End If

    End If
  Else
    m_bCompletedOk = False
  End If
  Me.Visible = False
End Sub

Private Sub Form_Load()
  If Not m_pExt Is Nothing Then
    'load fields
    If Not m_pExt.RoadLayer Is Nothing Then
      LoadFields m_pExt.RoadLayer.FeatureClass
      'select first element
      If cboFields.ListCount > -1 Then
        cboFields.ListIndex = 0
      End If
    End If
  End If
End Sub

Private Sub Form_Terminate()
  Set m_pExt = Nothing
End Sub

Private Sub optCreate_Click()
  'disable combo box
  cboFields.Enabled = False
  lblE.Enabled = False
  ckOverride.Enabled = False
```

```
  'enable text field
  tboName.Enabled = True
  tboName.SetFocus
  lblC.Enabled = True

End Sub

Private Sub optExists_Click()
  'disable test field
  tboName.Enabled = False
  lblC.Enabled = False
  'enable combo
  cboFields.Enabled = True
  lblE.Enabled = True
  ckOverride.Enabled = True
End Sub

Public Property Let RSAExtension(ByRef RSA As clsExt)
  Set m_pExt = RSA
End Property

Public Property Get CompletedOK() As Boolean
  CompletedOK = m_bCompletedOk
End Property

Private Sub LoadFields(pFClass As IFeatureClass)
  'load fields
  If Not pFClass Is Nothing Then
    Dim pFields As IFields
    Set pFields = pFClass.Fields
    Dim i As Integer
    For i = 0 To pFields.FieldCount - 1
      cboFields.AddItem pFields.Field(i).Name
    Next i
    Set pFields = Nothing
  End If
End Sub

Public Property Get OverrideValues() As Boolean
  If ckOverride.Enabled Then
    If ckOverride.Value = 1 Then
      OverrideValues = True
    Else
      OverrideValues = False
    End If
  Else: OverrideValues = True
  End If
End Property
```

FORM - frmOptions (frmOptions.frm)

```
Option Explicit
Private m_pApp As IApplication
Private m_pPar As clsExt

Public Sub SetUpBoxes(pPar As clsExt, pApp As IApplication)
```

```
  Set m_pApp = pApp
  Set m_pPar = pPar

  'find all available extensions that implement SedimentModel
  Dim pExtManager As IExtensionManager
  Set pExtManager = pApp
  Dim pExt As IExtension
  Dim i As Integer
  For i = 0 To pExtManager.ExtensionCount - 1
    Set pExt = pExtManager.Extension(i)
    If TypeOf pExt Is ISedimentModel Then
      lsbSedModel.AddItem pExt.Name
    End If
  Next i
  'identify the one stored within the parent extension and select it
  lsbSedModel.ListIndex = Util.FindItemInListBox(lsbSedModel,
m_pPar.SedModelName)
  'if no selection has been made select the first in the list
  If lsbSedModel.ListIndex = -1 And lsbSedModel.ListCount > 0 Then
    lsbSedModel.ListIndex = 0
  End If

  'populate combo boxes with values
  cboAge.AddItem "0"
  cboAge.AddItem "3"
  For i = 0 To 18
    cboGrade.AddItem i
  Next i
  cboSurface.AddItem "P"
  cboSurface.AddItem "DO"
  cboSurface.AddItem "G2"
  cboSurface.AddItem "G6"
  cboSurface.AddItem "N"
  For i = 1 To 24
    cboWidth.AddItem i
  Next i
  cboTraffic.AddItem "H"
  cboTraffic.AddItem "M"
  cboTraffic.AddItem "L"
  cboTraffic.AddItem "A"
  For i = 0 To 10
    cboCover.AddItem i * 10
  Next i

  'identify the values stored in extension and select them
  cboAge.ListIndex = Util.FindItemInComboBox(cboAge,
CStr(m_pPar.DefaultRoadAge))
  cboGrade.ListIndex = Util.FindItemInComboBox(cboGrade,
CStr(m_pPar.DefaultRoadGrade))
  cboSurface.ListIndex = Util.FindItemInComboBox(cboSurface,
m_pPar.DefaultRoadSurface)
  cboWidth.ListIndex = Util.FindItemInComboBox(cboWidth,
CStr(m_pPar.DefaultRoadWidth))
  cboTraffic.ListIndex = Util.FindItemInComboBox(cboTraffic,
m_pPar.DefaultRoadTraffic)
  cboCover.ListIndex = Util.FindItemInComboBox(cboCover,
CStr(m_pPar.DefaultSlopeCover))

End Sub
```

```
Private Sub btnCancel_Click()
  Unload Me
End Sub

Private Sub btnSet_Click()
  'get value from the contributing area field
  On Error GoTo erh

  'get value from the lsbSedModel
  m_pPar.SedModelName = lsbSedModel.List(lsbSedModel.ListIndex)

  m_pPar.DefaultRoadAge = Val(cboAge.List(cboAge.ListIndex))
  m_pPar.DefaultRoadGrade = Val(cboGrade.List(cboGrade.ListIndex))
  m_pPar.DefaultRoadSurface = cboSurface.List(cboSurface.ListIndex)
  m_pPar.DefaultRoadWidth = Val(cboWidth.List(cboWidth.ListIndex))
  m_pPar.DefaultRoadTraffic = cboTraffic.List(cboTraffic.ListIndex)
  m_pPar.DefaultSlopeCover = Val(cboCover.List(cboCover.ListIndex))

  Unload Me
  Exit Sub
erh:
  MsgBox "error in set button action" & Error
End Sub

Private Sub Form_Unload(Cancel As Integer)
  Set m_pApp = Nothing
  Set m_pPar = Nothing
End Sub
```

FORM – frmSetData (frmSetData.frm)

```
Option Explicit

Private m_iCurrentIndex As Integer
Private m_bSucces As Boolean
Private m_pParent As clsStart

Private Sub btnAdd_Click()
  PromoteLayer
End Sub

Private Sub btnBack_Click()
  If m_iCurrentIndex > 0 Then
    m_iCurrentIndex = m_iCurrentIndex - 1
    lsbIds.Selected(m_iCurrentIndex) = True
    lblHead.Caption = "Select " & lsbIds.List(m_iCurrentIndex) & "
Layer"

    If "" = lsbUserDecisions.List(m_iCurrentIndex) Then
      btnAdd.Enabled = True
      btnRemove.Enabled = False
    Else
      btnAdd.Enabled = False
      btnRemove.Enabled = True
    End If
```

```
    If m_iCurrentIndex = lsbIds.ListCount - 2 Then
      btnNext.Caption = "Next"
    ElseIf m_iCurrentIndex = 0 Then
      btnBack.Enabled = False
    End If
  End If

End Sub

Private Sub btnCancel_Click()
  'hide form
  Me.Hide
  'set flag unsucessful
  m_bSucces = False
End Sub

Private Sub btnNext_Click()
  If m_iCurrentIndex < lsbIds.ListCount - 1 Then
    m_iCurrentIndex = m_iCurrentIndex + 1
    If m_iCurrentIndex = lsbIds.ListCount - 1 Then
      btnNext.Caption = "Finish"
    End If
  Else
    'check if all layers have been set and send them to the extension
    If ValidateLayers() Then
      'send layers
      m_pParent.ReceiveLayers lsbUserDecisions.List(0),
lsbUserDecisions.List(1), _
                              lsbUserDecisions.List(2),
lsbUserDecisions.List(3), _
                              ckbCopy.Value
      'flag operation as succesful
      m_bSucces = True
      Me.Hide
    Else
      MsgBox "Please set all required layers!"
      'flag operation as insuccesful
      m_bSucces = False
    End If
  End If
  lsbIds.Selected(m_iCurrentIndex) = True
  lblHead.Caption = "Select " & lsbIds.List(m_iCurrentIndex) & " Layer"
  btnBack.Enabled = True
  If "" = lsbUserDecisions.List(m_iCurrentIndex) Then
    btnAdd.Enabled = True
    btnRemove.Enabled = False
  Else
    btnAdd.Enabled = False
    btnRemove.Enabled = True
  End If
End Sub

Private Sub btnRemove_Click()
  lsbAvailable.AddItem lsbUserDecisions.List(m_iCurrentIndex)
  lsbUserDecisions.List(m_iCurrentIndex) = ""

  btnAdd.Enabled = True
  btnRemove.Enabled = False
```

```
End Sub

Private Sub Form_Initialize()
  Me.lsbIds.AddItem "Roads", 0
  lsbUserDecisions.AddItem "", 0
  Me.lsbIds.AddItem "Streams", 1
  lsbUserDecisions.AddItem "", 0
  Me.lsbIds.AddItem "Culverts", 2
  lsbUserDecisions.AddItem "", 0
  Me.lsbIds.AddItem "DEM", 3
  lsbUserDecisions.AddItem "", 0

  m_iCurrentIndex = 0
  lsbIds.Selected(m_iCurrentIndex) = True
  'check by default
  'ckbCopy.Value = vbChecked
  ckbCopy.Enabled = False
End Sub
Public Sub AddLayers(pMap As IMap, ByRef pStartButt As clsStart)
  Set m_pParent = pStartButt
  Dim counter As Long
  For counter = 0 To pMap.LayerCount - 1
    lsbAvailable.AddItem pMap.Layer(counter).Name
  Next counter
End Sub
Private Function ValidateLayers() As Boolean
  Dim counter As Integer
  'will verify ONLY the required fields 0 - 3
  For counter = 0 To lsbUserDecisions.ListCount - 1
    If "" = lsbUserDecisions.List(counter) Then
      ValidateLayers = False
      Exit Function
    End If
  Next counter
  ValidateLayers = True
End Function

Public Property Get CompletedSuccessfuly() As Boolean
  CompletedSuccessfuly = m_bSucces
End Property

Private Sub lsbAvailable_DblClick()
  If True = btnAdd.Enabled Then PromoteLayer
End Sub

Private Sub PromoteLayer()
   If lsbAvailable.ListIndex > -1 Then
    lsbUserDecisions.List(m_iCurrentIndex) =
lsbAvailable.List(lsbAvailable.ListIndex)
    lsbAvailable.RemoveItem (lsbAvailable.ListIndex)

    btnAdd.Enabled = False
    btnRemove.Enabled = True
  End If
End Sub
```

MODULE - ConnectFnct (ConnectFnct.bas)

```
Option Explicit

'returns the value if unique positive, -1 if no children, -100 if
multiple children.
Public Function CheckUniqueChild(lChildren() As Long) As Long
  Dim i As Integer
  Dim iPosValues As Integer
  iPosValues = 0
  Dim lUniqueChild As Long
  lUniqueChild = -1

  For i = LBound(lChildren()) To UBound(lChildren())
    If lChildren(i) > 0 Then
      iPosValues = iPosValues + 1
      lUniqueChild = lChildren(i)
    End If
  Next i

  If iPosValues = 0 Then
    CheckUniqueChild = -1 'no children
    Exit Function
  ElseIf iPosValues = 1 Then 'unique child return value
    CheckUniqueChild = lUniqueChild
    Exit Function
  Else
    CheckUniqueChild = -100
  End If
End Function

'has to run inside edit session
Public Sub ReuniteChild(pSegments() As IFeature, lP1x As Long, lP2x As
Long, _
                        lP3x As Long, lChx As Long, lIdx As Long)

  'pSegments is an array that holds the parents first and the child as
last element
  Dim pChild As IFeature
  Set pChild = pSegments(UBound(pSegments))
  If Not pChild Is Nothing Then
    Dim i As Integer
    For i = 0 To UBound(pSegments) - 1
      pSegments(i).Value(lChx) = pChild.Value(lIdx)
      pSegments(i).Store
    Next i
    Dim pIndexes(2) As Long

    pIndexes(0) = lP1x
    pIndexes(1) = lP2x
    pIndexes(2) = lP3x
    Dim iStop As Integer
    iStop = 2
    If UBound(pSegments) - 1 < iStop Then iStop = UBound(pSegments) - 1
    For i = 0 To iStop
      pChild.Value(pIndexes(i)) = pSegments(i).Value(lIdx)
      pChild.Store
    Next i
  End If
```

```
  'release memory
  Set pChild = Nothing
End Sub

'will follow upstream from this segment setting all to the culvert this
segment drains to
Public Sub SetUpstream(ByRef lCulNo As Long, ByRef pRoad As
IFeatureLayer, ByRef pFeat As IFeature, ByRef lP1x As Long, _
                       ByRef lP2x As Long, ByRef lP3x As Long, ByRef lCx)
  'get parents
  Dim pParArray(2) As Long
  pParArray(0) = pFeat.Value(lP1x)
  pParArray(1) = pFeat.Value(lP2x)
  pParArray(2) = pFeat.Value(lP3x)

  Dim i As Integer
  For i = 0 To 2
    If pParArray(i) > 0 Then
      Dim pFilter As IQueryFilter
      Set pFilter = New QueryFilter
      pFilter.WhereClause = "RSAID = " & pParArray(i)
      Dim pPar As IFeature
      Dim pFeatCur As IFeatureCursor
      Set pFeatCur = pRoad.Search(pFilter, False)
      Set pPar = pFeatCur.NextFeature
      pPar.Value(lCx) = lCulNo
      pPar.Store
      SetUpstream lCulNo, pRoad, pPar, lP1x, lP2x, lP3x, lCx

      'release memory
      Set pFilter = Nothing
      Set pPar = Nothing
      Set pFeatCur = Nothing
    End If
  Next i
  End Sub

'changes the parent that indetifies to old parent to the given new
parent
Public Sub ChangeOneParent(pLayer As IFeatureLayer, lChildId As Long,
lOldParentId As Long, lNewParentId As Long, _
                           lP1x As Long, lP2x As Long, lP3x As Long,
lIdx As Long)
  'find the feature that is the given child lChildId
  Dim pFeatCur As IFeatureCursor
  Dim pFilter As IQueryFilter
  Set pFilter = New QueryFilter
  pFilter.WhereClause = "RSAID = " & lChildId
  Set pFeatCur = pLayer.Search(pFilter, False)
  Dim pChildFeat As IFeature
  Set pChildFeat = pFeatCur.NextFeature
  'find the parent that is lOldParent and change to lNewParent
  If Not pChildFeat Is Nothing Then
    If pChildFeat.Value(lP1x) = lOldParentId Then
      pChildFeat.Value(lP1x) = lNewParentId
      pChildFeat.Store
      Exit Sub
    ElseIf pChildFeat.Value(lP2x) = lOldParentId Then
```

```
      pChildFeat.Value(lP2x) = lNewParentId
      pChildFeat.Store
      Exit Sub
    ElseIf pChildFeat.Value(lP3x) = lOldParentId Then
      pChildFeat.Value(lP3x) = lNewParentId
      pChildFeat.Store
    End If
  End If

  'release memory
  Set pFeatCur = Nothing
  Set pFilter = Nothing
  Set pChildFeat = Nothing
End Sub
'returns false if no parents were found for this particular roadfeature
Public Function HasParents(pFeat As IFeature, lP1x As Long, lP2x As
Long, _
                           lP3x As Long) As Boolean
If pFeat.Value(lP1x) <> -1 Then
  HasParents = True
  Exit Function
ElseIf pFeat.Value(lP2x) <> -1 Then
  HasParents = True
  Exit Function
ElseIf pFeat.Value(lP3x) <> -1 Then
  HasParents = True
  Exit Function
End If
HasParents = False
End Function
'changes parents of this child to -1 and the child to no parents
Public Sub SevereChild(pChild As IFeature, pLayer As IFeatureLayer, _
                       lP1x, lP2x, lP3x, lChx)
  'HAS TO RUN INSIDE EDIT SESION
  Dim pFilter As IQueryFilter
  Set pFilter = New QueryFilter
  Dim pCursor As IFeatureCursor
  Dim pParent As IFeature
  Dim lIndexes(2) As Long
  lIndexes(0) = lP1x
  lIndexes(1) = lP2x
  lIndexes(2) = lP3x
  Dim i As Integer
  Dim par As Long
  For i = 0 To 2
    par = pChild.Value(lIndexes(i))
    If par <> -1 Then
      pFilter.WhereClause = "RSAID = " & par
      Set pCursor = pLayer.Search(pFilter, False)
      Set pParent = pCursor.NextFeature
      If Not pParent Is Nothing Then
        pParent.Value(lChx) = -1
        pParent.Store
      End If
    End If
  Next i

  pChild.Value(lP1x) = -1
  pChild.Value(lP2x) = -1
```

```
  pChild.Value(lP3x) = -1
  pChild.Store

  'clean up
  Set pFilter = Nothing
  Set pCursor = Nothing
  Set pParent = Nothing
End Sub
'return the an array of features that intersect with the point with the
upper segmnent
'as first element and the lower segment as the last
Public Function IdentifyUpperLower(pLayer As IFeatureLayer, pPoint As
IPoint, _
                                   lTpx As Long) As IFeature()
  On Error GoTo erh

  'create a selection set with all feature that touch with point
  Dim pUpperLower() As IFeature
  Dim pFilter As ISpatialFilter
  Set pFilter = New SpatialFilter
  Set pFilter.Geometry = pPoint
  pFilter.SpatialRel = esriSpatialRelTouches
  Dim pSelSet As ISelectionSet
  Dim pDataset As IDataset
  Set pDataset = pLayer
  Set pSelSet = pLayer.FeatureClass.Select(pFilter,
esriSelectionTypeHybrid, _
                              esriSelectionOptionNormal,
pDataset.Workspace)
  'get cursor into all these elements of the selection set
  Dim pCursor As IFeatureCursor
  pSelSet.Search Nothing, False, pCursor

  'fill up a collection of all these segments
  Dim colSegments As New Collection
  Dim pSwap As IFeature
  Set pSwap = pCursor.NextFeature
  Do While Not pSwap Is Nothing
    colSegments.Add pSwap
    'MsgBox "point touches " & pSwap.OID
    Set pSwap = pCursor.NextFeature
  Loop
  'find Segments that have pPoint as their geometric to point ( =
parents)
  Dim colPar As New Collection
  Dim pPolyline As IPolyline
  Dim pElem As Variant
  If colSegments.count > 1 Then
    For Each pElem In colSegments
      Set pSwap = pElem
      Set pPolyline = pSwap.Shape
      If Util.ComparePointLocations(pPolyline.ToPoint, pPoint) Then
        colPar.Add pSwap
        'MsgBox "parent " & pSwap.OID
      End If
    Next
  Else 'one element means it can only be the child
    ReDim pUpperLower(0)
    Set pUpperLower(0) = colSegments(1)
```

```
    IdentifyUpperLower = pUpperLower
    GoTo CleanUp
  End If
  'find the child with a query filter on the parents "TOPT"
  'all parents should flow to same "TOPT" if flow geometry
  'has been maintained correctly
  Dim lToPt As Long
  Set pSwap = colPar(1)
  lToPt = pSwap.Value(lTpx)
  Dim pQFilter As IQueryFilter
  Set pQFilter = New QueryFilter
  pQFilter.WhereClause = "FROMPT = " & lToPt
  'pSwap should now be the child
  pSelSet.Search pQFilter, False, pCursor
  Set pSwap = pCursor.NextFeature
  'fill in the upperlower array with the parents first and the child as
last element
  Dim i As Long
  i = colPar.count
  ReDim pUpperLower(i)
  'load child
  Set pUpperLower(i) = pSwap
  'MsgBox "loaded child " & pSwap.OID & " at " & i
  'load parents
  i = 0
  For Each pElem In colPar
    Set pSwap = pElem
    Set pUpperLower(i) = pSwap
    i = i + 1
  Next
  IdentifyUpperLower = pUpperLower
  Set pQFilter = Nothing
CleanUp:
  Set pSwap = Nothing
  Set pPolyline = Nothing
  Set pFilter = Nothing
  Set pCursor = Nothing
  Set colSegments = Nothing
  Set colPar = Nothing
  Set pSelSet = Nothing
  Set pDataset = Nothing
  Exit Function
erh:
  MsgBox "error in identify upper/lower " & Error
End Function

Public Function SumUpSed(pFeatClass As IFeatureClass, lSedFieldIndex As
Long, _
                         sCulFieldName As String, lCulvert As Long) As
Double
  On Error GoTo erh

  Dim dTotalSed As Double
  Dim pFilter As IQueryFilter
  Set pFilter = New QueryFilter
  pFilter.WhereClause = sCulFieldName & " = " & lCulvert
  Dim pCursor As IFeatureCursor
  Set pCursor = pFeatClass.Search(pFilter, False)
  Dim pFeat As IFeature
```

```
  Set pFeat = pCursor.NextFeature
  Do While Not pFeat Is Nothing
    dTotalSed = dTotalSed + pFeat.Value(lSedFieldIndex)
    Set pFeat = pCursor.NextFeature
  Loop

  SumUpSed = dTotalSed
  Exit Function

  'release memory
  Set pFilter = Nothing
  Set pCursor = Nothing
  Set pFeat = Nothing
erh:
  MsgBox "error in SumUpSed " & Error
End Function

Public Sub SumAttrib(pReceiver As IFeature, pContributor As IFeature,
lFieldIndex As Long)
  Dim dValue As Double
  dValue = pReceiver.Value(lFieldIndex)
  pReceiver.Value(lFieldIndex) = dValue +
pContributor.Value(lFieldIndex)
End Sub

Public Sub SplitAttrib(pReceiver As IFeature, pOriginal As IFeature,
lFieldIndex As Long)
  'Receiver is blank. has no value for this attrib
  'orig will receive the difference after splitting attrib
  Dim pRecPoly As IPolyline, pOrigPoly As IPolyline
  Set pRecPoly = pReceiver.Shape
  Set pOrigPoly = pOriginal.Shape
  Dim dVal As Double
  dVal = pOriginal.Value(lFieldIndex)
  pReceiver.Value(lFieldIndex) = dVal * pRecPoly.Length /
pOrigPoly.Length
  pOriginal.Value(lFieldIndex) = dVal - pReceiver.Value(lFieldIndex)

  'release memory
  Set pRecPoly = Nothing
  Set pOrigPoly = Nothing
End Sub
'inserts a node in the FROMPT or TOPT field of an RSA road table
'if the given point is on one of the line ends
Public Function DisconnectNodes(pLineFeature As IFeature, _
                                pPointFeature As IFeature, _
                                lFpx As Long, _
                                lTpx As Long, _
                                lMaxPointId As Long) As Long
  Dim pPolyline As IPolyline
  Dim pCompPoint As IPoint
  Dim pEndPoint As IPoint

  'identify end that touches
  Set pPolyline = pLineFeature.Shape
  Set pCompPoint = pPointFeature.Shape
  'try the "from" end first
  Set pEndPoint = pPolyline.FromPoint
  If Util.ComparePointLocations(pCompPoint, pEndPoint) Then
```

```
    pLineFeature.Value(lFpx) = lMaxPointId + 1 ' set flow "FROM" to this
point
    pLineFeature.Store
    lMaxPointId = lMaxPointId + 1
  End If
  'try the "to" end
  Set pEndPoint = pPolyline.ToPoint
  If Util.ComparePointLocations(pCompPoint, pEndPoint) Then
    pLineFeature.Value(lTpx) = lMaxPointId + 1 ' set flow "TO" to this
point
    pLineFeature.Store
    lMaxPointId = lMaxPointId + 1
  End If

  DisconnectNodes = lMaxPointId
End Function
```

MODULE - EnforceFnct (EnforceFnct.bas)

```
Option Explicit

'call this within an edit session
Public Sub SimplifyPaths(pFeatClass As IFeatureClass)
  On Error GoTo erh

  Dim pCursor As IFeatureCursor
  Set pCursor = pFeatClass.Search(Nothing, False)
  Dim pFeat As IFeature
  Set pFeat = pCursor.NextFeature
  Dim pPolyline As IPolyline
  Dim pGeoColl As IGeometryCollection
  Dim pNewFeat As IFeature
  Dim pNewPolyline As IPolyline
  Dim pNewGeoColl As IGeometryCollection
  Dim i As Integer

  Do While Not pFeat Is Nothing
    Set pPolyline = pFeat.Shape
    Set pGeoColl = pPolyline
    If pGeoColl.GeometryCount > 1 Then
      For i = 0 To pGeoColl.GeometryCount - 1
        'create a new polyline from each path
        Set pNewPolyline = New Polyline
        Set pNewGeoColl = pNewPolyline
        pNewGeoColl.AddGeometry pGeoColl.Geometry(i)
        'create a polyline for each path
        Set pNewFeat = pFeatClass.CreateFeature
        Util.CopyAllAtributes pFeat, pNewFeat
        Set pNewFeat.Shape = pNewPolyline
        pNewFeat.Store
      Next i
      'remove the initial polyline
      pFeat.Delete
    End If
    Set pFeat = pCursor.NextFeature
  Loop
```

```
  'release memory
  Set pCursor = Nothing
  Set pFeat = Nothing
  Set pPolyline = Nothing
  Set pGeoColl = Nothing
  Set pNewFeat = Nothing
  Set pNewPolyline = Nothing
  Set pNewGeoColl = Nothing

  Exit Sub
erh:
  MsgBox "error in SimplifyPaths " & Error
End Sub

'run this within an edit session
Public Sub ForceEndConnectivity(pFeatClass As IFeatureClass)
  On Error GoTo erh

  Dim pCursor As IFeatureCursor
  Set pCursor = pFeatClass.Search(Nothing, False)
  Dim pFeat As IFeature
  Set pFeat = pCursor.NextFeature
  Dim pPolyline As IPolyline
  Dim pSpFilter As ISpatialFilter
  Set pSpFilter = New SpatialFilter
  pSpFilter.SpatialRel = esriSpatialRelWithin
  Dim pSplitCursor As IFeatureCursor

  Do While Not pFeat Is Nothing
    Set pPolyline = pFeat.Shape

    Set pSpFilter.Geometry = pPolyline.FromPoint
    Set pSplitCursor = pFeatClass.Search(pSpFilter, False)
    SplitSeachAtPoint pSplitCursor, pPolyline.FromPoint, pFeatClass

    Set pSpFilter.Geometry = pPolyline.ToPoint
    Set pSplitCursor = pFeatClass.Search(pSpFilter, False)
    SplitSeachAtPoint pSplitCursor, pPolyline.ToPoint, pFeatClass

    Set pFeat = pCursor.NextFeature
  Loop

  Exit Sub
erh:
  MsgBox "error in ForceEndConnectivity " & Error
End Sub

'Run inside edit session
Private Sub SplitSeachAtPoint(pCursor As IFeatureCursor, pPoint As
IPoint, _
                              pFClass As IFeatureClass)
  On Error GoTo erh

  Dim pFeat As IFeature
  Set pFeat = pCursor.NextFeature
  Dim pPieces(1) As IPolyline
  Dim pNewFeat As IFeature
  Do While Not pFeat Is Nothing
    Util.CutPolylineAtPoint pFeat.Shape, pPoint, pPieces
```

```
    'create new features
    If Not pPieces(0) Is Nothing Then
      Set pNewFeat = pFClass.CreateFeature
      Util.CopyAllAtributes pFeat, pNewFeat
      Set pNewFeat.Shape = pPieces(0)
      pNewFeat.Store
    End If
    If Not pPieces(1) Is Nothing Then
      Set pNewFeat = pFClass.CreateFeature
      Util.CopyAllAtributes pFeat, pNewFeat
      Set pNewFeat.Shape = pPieces(1)
      pNewFeat.Store
    End If
    'delete original feature
    pFeat.Delete
    'iterate
    Set pFeat = pCursor.NextFeature
  Loop
  'release memory
  Set pFeat = Nothing
  Set pNewFeat = Nothing

  Exit Sub
erh:
  MsgBox "error in SplitSeachAtpoint " & Error
End Sub
```

MODULE - ErrorHandling (ErrorHandling.bas)

```
Option Explicit
'
' FILE AUTOMATICALLY GENERATED BY ESRI ERROR HANDLER ADDIN
' DO NOT EDIT OR REMOVE THIS FILE FROM THE PROJECT
'
Dim pErrorLog As New ErrorHandlerUI.ErrorDialog


Private Sub DisplayVersion2Dialog(sProcedureName As String,
sErrDescription As String)
  Beep
  MsgBox "An error has occured in the application.   Record the call
stack sequence" & vbCrLf & "and the description of the error." & vbCrLf
& vbCrLf & _
         "Error Call Stack Sequence " & vbCrLf & vbTab & sProcedureName
& vbCrLf & sErrDescription, vbExclamation + vbOKOnly, "Unexpected
Program Error"
End Sub

Private Sub DisplayVersion3Dialog(sProcedureName As String,
sErrDescription As String, parentHWND As Long, raiseException As
Boolean)
  Beep
  MsgBox "An error has occured in the application.   Record the call
stack sequence" & vbCrLf & "and the description of the error." & vbCrLf
& vbCrLf & _
         "Error Call Stack Sequence " & vbCrLf & vbTab & sProcedureName
& vbCrLf & sErrDescription, vbExclamation + vbOKOnly, "Unexpected
Program Error"
```

```
End Sub

Private Sub DisplayVersion4Dialog(sProcedureName As String,
sErrDescription As String, parentHWND As Long)
  pErrorLog.AppendErrorText "Record Call Stack Sequence - Bottom line is
error line." & vbCrLf & vbCrLf & vbTab & sProcedureName & vbCrLf &
sErrDescription
  pErrorLog.Visible = True


End Sub

Public Sub HandleError(ByVal bTopProcedure As Boolean, _
                       ByVal sProcedureName As String, _
                       ByVal lErrNumber As Long, _
                       ByVal sErrSource As String, _
                       ByVal sErrDescription As String, _
                       Optional ByVal version As Long = 1, _
                       Optional ByVal parentHWND As Long = 0, _
                       Optional ByVal reserved1 As Variant = 0, _
                       Optional ByVal reserved2 As Variant = 0, _
                       Optional ByVal reserved3 As Variant = 0)
  ' Generic Error handling Function - This function should be called
with
  ' the following Arguments
  '
  ' Boolean    -in-  True if called from a top level procedure - Event /
Method / Property
  ' String     -in-  Name of function called from
  ' Long       -in-  Error Number (retrieved from Err object)
  ' String     -in-  Error Source (retrieved from Err object)
  ' String     -in-  Error Description (retrieved from Err object)
  ' Long       -in-  Version of Function (optional Default 1)
  ' parentHWND -in-  Parent Hwnd for error dialogs, NULL is valid
  ' reserved1  -in-
  ' reserved2  -in-
  ' reserved3  -in-


  ' Clear the error object
  Err.Clear

  ' Static variable used to control the call stack formatting
  Static entered As Boolean

  If (bTopProcedure) Then
    ' Top most procedure in call stack so report error to user
    ' Via a dialog
    If (Not entered) Then
      sErrDescription = vbCrLf & "Error Number " & vbCrLf & vbTab &
CStr(lErrNumber) & vbCrLf & "Description" & vbCrLf & vbTab &
sErrDescription & vbCrLf & vbCrLf
    End If
    entered = False
    If (version = 4) Then
      DisplayVersion4Dialog sProcedureName, sErrDescription, parentHWND
    ElseIf (version = 3) Then
      Dim raiseError As Boolean
```

```
      DisplayVersion3Dialog sProcedureName, sErrDescription, parentHWND,
raiseError
      If (raiseError) Then Err.Raise lErrNumber, sErrSource, vbTab &
sProcedureName & vbCrLf & sErrDescription
    ElseIf (version = 2) Then
      DisplayVersion2Dialog sProcedureName, sErrDescription
    Else
      Beep
      MsgBox "An error has occured in the application.   Record the call
stack sequence" & vbCrLf & "and the description of the error." & vbCrLf
& vbCrLf & _
           "Error Call Stack Sequence " & vbCrLf & vbTab &
sProcedureName & vbCrLf & sErrDescription, vbExclamation + vbOKOnly,
"Unexpected Program Error"
    End If
  Else
    ' An error has occured but we are not at the top of the call stack
    ' so append the callstack and raise another error
    If (Not entered) Then sErrDescription = vbCrLf & "Error Number " &
vbCrLf & vbTab & CStr(lErrNumber) & vbCrLf & "Description" & vbCrLf &
vbTab & sErrDescription & vbCrLf & vbCrLf
    entered = True
    Err.Raise lErrNumber, sErrSource, vbTab & sProcedureName & vbCrLf &
sErrDescription
  End If
End Sub

Public Function GetErrorLineNumberString(ByVal lLineNumber As Long) As
String
  ' Test the line number if it is non zero create a string
  If (lLineNumber <> 0) Then GetErrorLineNumberString = "Line : " &
lLineNumber
End Function
```

MODULE - Util (RSAUtil.bas)

```
Option Explicit

Public Function FindOneFeature(pFeatClass As IFeatureClass, lFName As
String, _
                               lValue As Long) As IFeature
  Dim pFilter As IQueryFilter
  Set pFilter = New QueryFilter
  pFilter.WhereClause = lFName & " = " & lValue
  Dim pCursor As IFeatureCursor
  Set pCursor = pFeatClass.Search(pFilter, False)
  Set FindOneFeature = pCursor.NextFeature

  Set pFilter = Nothing
  Set pCursor = Nothing
End Function

Public Function CreateRSAFields(pRoadClass As IFeatureClass, _
                                pCulvClass As IFeatureClass) As Boolean
  Dim pField As IField
  Dim pFieldEdit As IFieldEdit
  On Error GoTo ErrorHandler
```

```
  Set pField = New Field
  Set pFieldEdit = pField

  With pFieldEdit
    'integer fields
    .Name = "RSAID"
    .Type = esriFieldTypeInteger
    AddNonExistingField pField, pRoadClass
    AddNonExistingField pField, pCulvClass

    .Name = "FROMPT"
    AddNonExistingField pField, pRoadClass

    .Name = "TOPT"
    AddNonExistingField pField, pRoadClass

    .Name = "PAR1"
    AddNonExistingField pField, pRoadClass

    .Name = "PAR2"
    AddNonExistingField pField, pRoadClass

    .Name = "PAR3"
    AddNonExistingField pField, pRoadClass

    .Name = "CHILD"
    AddNonExistingField pField, pRoadClass

    .Name = "CULV"
    AddNonExistingField pField, pRoadClass

    .Name = "REMCD"
    AddNonExistingField pField, pCulvClass

    'fields of type double
    .Type = esriFieldTypeDouble
    .Name = "SEDPROD"
    AddNonExistingField pField, pRoadClass

    .Name = "DELPOT"
    AddNonExistingField pField, pCulvClass

    .Name = "SED"
    AddNonExistingField pField, pCulvClass
  End With
  CreateRSAFields = True
  Exit Function

ErrorHandler:
  MsgBox "if editor is on please turn off first"
  CreateRSAFields = False
End Function

Public Sub AddNonExistingField(pNewField As IField, pFeatureClass As
IFeatureClass)
  Dim pos As Long
  pos = pFeatureClass.FindField(pNewField.Name)
  If pos > -1 Then
    Dim pOldField As IField
```

```
    Set pOldField = pFeatureClass.Fields.Field(pos)
    pFeatureClass.DeleteField pOldField
  End If
  pFeatureClass.AddField pNewField
End Sub

Public Function GetFieldIndexes(pFeatClass As IFeatureClass, sType As
String) As Long()
  Dim lIndexes() As Long
  If StrComp(sType, "road", vbTextCompare) = 0 Then
    ReDim lIndexes(10)
    lIndexes(0) = pFeatClass.FindField("RSAID")
    lIndexes(1) = pFeatClass.FindField("ZFROM")
    lIndexes(2) = pFeatClass.FindField("ZTO")
    lIndexes(3) = pFeatClass.FindField("FROMPT")
    lIndexes(4) = pFeatClass.FindField("TOPT")
    lIndexes(5) = pFeatClass.FindField("PAR1")
    lIndexes(6) = pFeatClass.FindField("PAR2")
    lIndexes(7) = pFeatClass.FindField("PAR3")
    lIndexes(8) = pFeatClass.FindField("CHILD")
    lIndexes(9) = pFeatClass.FindField("CULV")
    lIndexes(10) = pFeatClass.FindField("SEDPROD")
  ElseIf StrComp(sType, "culvert", vbTextCompare) = 0 Then
    ReDim lIndexes(3)
    lIndexes(0) = pFeatClass.FindField("RSAID")
    lIndexes(1) = pFeatClass.FindField("REMCD")
    lIndexes(2) = pFeatClass.FindField("DELPOT")
    lIndexes(3) = pFeatClass.FindField("SED")
  End If
  GetFieldIndexes = lIndexes
End Function
'returns a containing the cell value.
'if anything goes wrong returns nodata.
Public Function GetCellValue(pRasterLayer As IRasterLayer, pPoint As
IPoint) As String
  Dim pRIDObj As IRasterIdentifyObj
  Dim pIdentify As IIdentify
  Dim pIDArray As IArray
  Dim pNewPoint As IPoint
  Set pNewPoint = New Point
  pNewPoint.X = pPoint.X
  pNewPoint.Y = pPoint.Y
  Set pIdentify = pRasterLayer
  Set pIDArray = pIdentify.Identify(pNewPoint)
  If Not pIDArray Is Nothing Then
    Set pRIDObj = pIDArray.Element(0)
    GetCellValue = pRIDObj.Name
  Else
    GetCellValue = "NoData"
  End If
  'clean up
  Set pNewPoint = Nothing
  Set pIDArray = Nothing
  Set pIdentify = Nothing
  Set pRIDObj = Nothing
End Function
'returns an array of two polylines with the upper segment at position 0
Public Function CutPolylineAtPoint(pInputLine As IPolyline, pSplitPoint
As IPoint, _
```

```
                                    ByRef pOutputLines() As IPolyline)
  Dim dDistAlong As Double
  Dim dDistFrom As Double
  Dim pPoint As IPoint

  pInputLine.QueryPointAndDistance esriNoExtension, pSplitPoint, False,
pPoint, _
                                     dDistAlong, dDistFrom, False
  'the segment that starts at "FROM" goes at index 0
  pInputLine.GetSubcurve 0#, dDistAlong, False, pOutputLines(0)
  'the segment that ends at "TO" goes at index 1
  pInputLine.GetSubcurve dDistAlong, pInputLine.Length, False,
pOutputLines(1)

  Set pPoint = Nothing
End Function

Public Function GetMaxValue(pFClass As IFeatureClass, lIdx As Long) As
Long
  'select all
  Dim pCursor As IFeatureCursor
  Set pCursor = pFClass.Search(Nothing, False)
  'go through all
  Dim pFeat As IFeature
  Dim lValue As Long, lMaxValue As Long
  lMaxValue = -2147483648#
  Set pFeat = pCursor.NextFeature
  Do While Not pFeat Is Nothing
    lValue = pFeat.Value(lIdx)
    If lValue > lMaxValue Then
      lMaxValue = lValue
    End If
  Set pFeat = pCursor.NextFeature
  Loop
  GetMaxValue = lMaxValue

  'release memory
  Set pCursor = Nothing
  Set pFeat = Nothing
End Function

Public Function GetMaxOfFields(pFClass As IFeatureClass, lIdx1 As Long,
lIdx2 As Long) As Long
  'get the maximum existing point number in the road table
  Dim lMaxField As Long
  Dim lMax1 As Long
  lMax1 = GetMaxValue(pFClass, lIdx1)
  'when there are no features set the lCurId to 1
  If lMax1 = -2147483648# Then
    lMax1 = 0
  End If
  Debug.Print "max FP id is" & lMax1
  Dim lMax2 As Long
  lMax2 = GetMaxValue(pFClass, lIdx2)
  If lMax2 = -2147483648# Then
    lMax2 = 0
  End If
  Debug.Print "max TP id is" & lMax2
  If lMax1 >= lMax2 Then
```

```
    lMaxField = lMax1
  Else: lMaxField = lMax2
  End If
  Debug.Print "max point id is " & lMaxField
  GetMaxOfFields = lMaxField
End Function
'changes the geometry of the given point feature according to
'the rules of the given snap agent
'has to run inside an edit session
Public Sub MovePointFeatToSnapLocation(pPointFeat As IFeature, _
                                       pSnapAgent As ISnapAgent, _
                                       tolerance As Double)
  Dim pPoint As IPoint
  Set pPoint = pPointFeat.Shape

  'MsgBox "trying to snap culvert"

  pSnapAgent.Snap Nothing, pPoint, tolerance

  'MsgBox "snapped successfuly"

  Set pPointFeat.Shape = pPoint
  pPointFeat.Store

End Sub
Public Function ComparePointLocations(pPoint1 As IPoint, pPoint2 As
IPoint) As Boolean
  Dim c_tol As Double
  c_tol = 0.000000001
  If (Math.Abs(pPoint1.X - pPoint2.X) <= c_tol) And (Math.Abs(pPoint1.Y
- pPoint2.Y) <= c_tol) Then
    ComparePointLocations = True
    Exit Function
  End If
  ComparePointLocations = False
End Function

Public Function VerifyName(pLayer As ILayer) As String
  If pLayer Is Nothing Then
    VerifyName = ""
    Exit Function
  End If
  VerifyName = pLayer.Name
End Function

Public Function ExistsLayer(sName As String, ByRef pLayer As ILayer, _
                            pDoc As IMxDocument) As Boolean
  Dim i As Integer
  For i = 0 To pDoc.FocusMap.LayerCount - 1
    If StrComp(sName, pDoc.FocusMap.Layer(i).Name, vbBinaryCompare) = 0
Then
      Set pLayer = pDoc.FocusMap.Layer(i)
      ExistsLayer = True
      Exit Function
    End If
  Next i
  Set pLayer = Nothing
  ExistsLayer = False
End Function
```

```
Public Sub AddRasterLayerToComboBox(cboBox As ComboBox, pMap As IMap)
  On Error GoTo erh
  cboBox.Clear
  Dim iLyrIndex As Long
  Dim pLyr As ILayer
  ' Add raster layers into  Combobox
  Dim iLayerCount As Integer
  iLayerCount = pMap.LayerCount
  If iLayerCount > 0 Then
    cboBox.Enabled = True
    For iLyrIndex = 0 To iLayerCount - 1
      Set pLyr = pMap.Layer(iLyrIndex)
      If (TypeOf pLyr Is IRasterLayer) Then
        cboBox.AddItem pLyr.Name
        cboBox.ItemData(cboBox.ListCount - 1) = iLyrIndex
      End If
    Next iLyrIndex
    If (cboBox.ListCount > 0) Then
      cboBox.ListIndex = 0
      cboBox.Text = pMap.Layer(cboBox.ItemData(0)).Name
    End If
  End If
  Exit Sub
erh:
  MsgBox "Add Raster Layer to ComboBox:" & Err.Description
End Sub

Public Sub AddFeatureLayerToComboBox(cboBox As ComboBox, pMap As IMap)
  On Error GoTo erh
  cboBox.Clear
  Dim iLyrIndex As Long
  Dim pLyr As ILayer
  ' Add feature layers into  Combobox
  Dim iLayerCount As Integer
  iLayerCount = pMap.LayerCount
  If iLayerCount > 0 Then
    cboBox.Enabled = True
    For iLyrIndex = 0 To iLayerCount - 1
      Set pLyr = pMap.Layer(iLyrIndex)
      If (TypeOf pLyr Is IFeatureLayer) Then
        cboBox.AddItem pLyr.Name
        cboBox.ItemData(cboBox.ListCount - 1) = iLyrIndex
      End If
    Next iLyrIndex
    If (cboBox.ListCount > 0) Then
      cboBox.ListIndex = 0
      cboBox.Text = pMap.Layer(cboBox.ItemData(0)).Name
    End If
  End If
  Exit Sub
erh:
  MsgBox "Add Feature Layer to ComboBox:" & Err.Description
End Sub

Public Function AddInputFromGxBrowser(cboInput As ComboBox, frm As Form,
_
                                      FilterRaster As Boolean) As
IDataset
```

```
  On Error GoTo erh
  Dim pGxObject As IGxObject
  Dim pFilter As IGxObjectFilter
  Dim pMiniBrowser As IGxDialog
  Dim pEnumGxObject As IEnumGxObject

  Set pMiniBrowser = New GxDialog
  If FilterRaster Then
    Set pFilter = New GxFilterRasterDatasets
  Else
    Set pFilter = New GxFilterFeatureClasses
  End If

  Set pMiniBrowser.ObjectFilter = pFilter
  pMiniBrowser.Title = "Select Dataset"

  If (pMiniBrowser.DoModalOpen(frm.hwnd, pEnumGxObject)) Then
    Set pGxObject = pEnumGxObject.Next
    Dim pGxDataset As IGxDataset
    Set pGxDataset = pGxObject
    Dim pDataset As esriCore.IDataset
    Set pDataset = pGxDataset.Dataset
    cboInput.Clear
    cboInput.AddItem pDataset.Name
    cboInput.ItemData(cboInput.ListCount - 1) = -1 ' -1 indicates
browsed data, not map layer
    cboInput.ListIndex = 0
  End If
  Set AddInputFromGxBrowser = pDataset
  ' move the focus off this command so OK or Cancel can be default if
'Enter' hit
'  Dim c As Control
'  For Each c In frm.Controls
'    If (c.TabIndex = cboInput.TabIndex + 1) Then
'      c.SetFocus
'      Exit For
'    End If
'  Next
    Set pGxObject = Nothing
    Set pMiniBrowser = Nothing
  Exit Function
erh:
    MsgBox "AddInputfromBrowser:" & Err.Description
End Function

Public Function FindFeatureXPoint(pFeatClass As IFeatureClass, pPoint As
IPoint) As IFeature
  Dim pSpatialFilter As ISpatialFilter
  Dim pCursor As IFeatureCursor
  Set pSpatialFilter = New SpatialFilter
  Set pSpatialFilter.Geometry = pPoint
  pSpatialFilter.SpatialRel = esriSpatialRelIntersects
  Set pCursor = pFeatClass.Search(pSpatialFilter, False)
  Set FindFeatureXPoint = pCursor.NextFeature

  Set pCursor = Nothing
  Set pSpatialFilter = Nothing
End Function
```

```
Public Sub CopyAllAtributes(pOriginal As IFeature, pCopy As IFeature)
  Dim pFields As IFields
  Set pFields = pOriginal.Fields
  Dim i As Integer
  Dim fieldIndex As Long
  Dim fieldName As String
  For i = 0 To pFields.FieldCount - 1
    If pFields.Field(i).Editable Then
      fieldIndex = pFields.FindField(pFields.Field(i).Name)
      pCopy.Value(fieldIndex) = pOriginal.Value(fieldIndex)
      pCopy.Store
    End If
  Next i
End Sub

Public Sub CopyAllNonRSAtributes(pOriginal As IFeature, pCopy As
IFeature, _
                                 lRSAIndexes() As Long, shapeName As
String)
  Dim pFields As IFields
  Set pFields = pOriginal.Fields
  Dim i As Integer, j As Integer
  Dim fieldIndex As Long
  Dim fieldName As String
  For i = 0 To pFields.FieldCount - 1
    If Not IsInArray(i, lRSAIndexes) And _
    StrComp(pFields.Field(i).Name, shapeName, vbBinaryCompare) <> 0 Then

      If pFields.Field(i).Editable Then
        fieldIndex = pFields.FindField(pFields.Field(i).Name)
        pCopy.Value(fieldIndex) = pOriginal.Value(fieldIndex)
        pCopy.Store
      End If

    End If
  Next i
End Sub
Private Function IsInArray(lValue As Integer, lComp() As Long) As
Boolean
  Dim i As Integer
  For i = LBound(lComp) To UBound(lComp)
    If lComp(i) = lValue Then
      IsInArray = True
      Exit Function
    End If
  Next i
  IsInArray = False
End Function

Public Function FindAllFeaturesXPoint(pFeatClass As IFeatureClass,
pPoint As IPoint) As IFeature()
  On Error GoTo erh
  Dim pSpatialFilter As ISpatialFilter
  Dim pCursor As IFeatureCursor
  Set pSpatialFilter = New SpatialFilter
  Set pSpatialFilter.Geometry = pPoint
  pSpatialFilter.SpatialRel = esriSpatialRelIntersects
  Set pCursor = pFeatClass.Search(pSpatialFilter, False)
  Dim pFeature As IFeature
```

```
  ReDim pFeat(0) As IFeature
  Set pFeature = pCursor.NextFeature
  Set pFeat(0) = pFeature
  Set pFeature = pCursor.NextFeature
  Do While Not pFeature Is Nothing
    ReDim Preserve pFeat(UBound(pFeat) + 1)
    Set pFeat(UBound(pFeat)) = pFeature
    Set pFeature = pCursor.NextFeature
  Loop
  FindAllFeaturesXPoint = pFeat

  Set pCursor = Nothing
  Set pSpatialFilter = Nothing
  Exit Function
erh:
  MsgBox "error in find all features x point " & Error
End Function

Public Function FindAllFeaturesNearPoint(pFeatureClass As IFeatureClass,
_
                                  pPoint As IPoint, dtolerance As Double)
As IFeature()
  On Error GoTo erh

  Dim pSpatialFilter As ISpatialFilter
  Dim pCursor As IFeatureCursor
   'expand point's envelope
  Dim pEnv As IEnvelope
  Set pEnv = pPoint.Envelope
  pEnv.Expand 1, 1, False
  pEnv.Expand dtolerance, dtolerance, True

  Set pSpatialFilter = New SpatialFilter
  Set pSpatialFilter.Geometry = pEnv
  pSpatialFilter.SpatialRel = esriSpatialRelIntersects
  Set pCursor = pFeatureClass.Search(pSpatialFilter, False)
  Dim pFeature As IFeature
  ReDim pFeat(0) As IFeature
  Set pFeature = pCursor.NextFeature
  Set pFeat(0) = pFeature
  Set pFeature = pCursor.NextFeature
  Do While Not pFeature Is Nothing
    ReDim Preserve pFeat(UBound(pFeat) + 1)
    Set pFeat(UBound(pFeat)) = pFeature
    Set pFeature = pCursor.NextFeature
  Loop
  FindAllFeaturesNearPoint = pFeat

  Set pCursor = Nothing
  Set pSpatialFilter = Nothing
  Set pEnv = Nothing

  Exit Function
erh:
  MsgBox "error in find all features near point " & Error
End Function

Public Function MergePolylines(pBase As IPolyline, pAppendice As
IPolyline) As IPolyline
```

```
  Dim pTopoOp As ITopologicalOperator
  Set pTopoOp = pBase
  Set MergePolylines = pTopoOp.Union(pAppendice)
End Function

Public Function FindFeatureNearPoint(pFeatClass As IFeatureClass, pPoint
As IPoint, _
                                     dtolerance As Double) As IFeature
  Dim pSpatialFilter As ISpatialFilter
  Dim pCursor As IFeatureCursor
  'expand point's envelope
  Dim pEnv As IEnvelope
  Set pEnv = pPoint.Envelope
  pEnv.Expand 1, 1, False
  pEnv.Expand dtolerance, dtolerance, True

  Set pSpatialFilter = New SpatialFilter
  Set pSpatialFilter.Geometry = pEnv
  pSpatialFilter.SpatialRel = esriSpatialRelIntersects
  Set pCursor = pFeatClass.Search(pSpatialFilter, False)
  Set FindFeatureNearPoint = pCursor.NextFeature

  Set pCursor = Nothing
  Set pSpatialFilter = Nothing
  Set pEnv = Nothing
End Function

Public Function ExistsField(pFClass As IFeatureClass, sName As String)
As Boolean
  If Not pFClass Is Nothing Then
    Dim pFields As IFields
    Set pFields = pFClass.Fields
    Dim i As Integer
    For i = 0 To pFields.FieldCount - 1
      If StrComp(sName, pFields.Field(i).Name, vbBinaryCompare) = 0 Then
        ExistsField = False
        Exit Function
      End If
    Next i
    Set pFields = Nothing
  End If
  ExistsField = True
End Function

Public Function AddField(pFClass As IFeatureClass, sName As String, _
                         fType As esriFieldType) As Boolean
  On Error GoTo erh
  Dim pField As IFieldEdit
  Set pField = New Field
  With pField
    pField.Name = sName
    pField.Type = fType
  End With
  pFClass.AddField pField
  AddField = True
  Exit Function
erh:
  AddField = False
  MsgBox "error in Util.AddField " & Error
```

```
End Function

Public Function SumValuesOnField(pFClass As IFeatureClass, lFIndex As
Long) As Double
  Dim dSum As Double
  Dim pCur As IFeatureCursor
  Set pCur = pFClass.Search(Nothing, False)
  Dim pFeat As IFeature
  Set pFeat = pCur.NextFeature
  Do While Not pFeat Is Nothing
    dSum = dSum + pFeat.Value(lFIndex)
    Set pFeat = pCur.NextFeature
  Loop
  SumValuesOnField = dSum
  Set pCur = Nothing
  Set pFeat = Nothing
End Function

Public Sub BreakPolyIntoPolySegments(pInPoly As IPolyline, pGeoColl As
IGeometryCollection)
  On Error GoTo erh
  Dim pPathColl As IGeometryCollection
  Set pPathColl = pInPoly
  Dim newPolyline As IPolyline
  Dim pNewPolyColl As IGeometryCollection
  Dim i As Integer
  For i = 0 To pPathColl.GeometryCount - 1
    Set newPolyline = New Polyline
    Set pNewPolyColl = newPolyline
    pNewPolyColl.AddGeometry pPathColl.Geometry(i)
    pGeoColl.AddGeometry newPolyline
  Next i

  Set pPathColl = Nothing
  Set newPolyline = Nothing
  Set pNewPolyColl = Nothing
  Exit Sub
erh:
  MsgBox "error in BreakPolyIntoPolySegments: " & Error
End Sub

Public Function CreateBoundarySnapAgent(pFClass As IFeatureClass) As
IFeatureSnapAgent
  Dim pFSnap As IFeatureSnapAgent
  Set pFSnap = New FeatureSnap
  With pFSnap
    Set .FeatureClass = pFClass
    .HitType = esriGeometryPartBoundary
  End With
  Set CreateBoundarySnapAgent = pFSnap
End Function

Public Function FindItemInListBox(lsbListBox As ListBox, _
                                  queryElem As String) As Integer
  Dim i As Integer
  For i = 0 To lsbListBox.ListCount - 1
    If StrComp(queryElem, lsbListBox.List(i)) = 0 Then
      FindItemInListBox = i
      Exit Function
```

```
    End If
  Next i
  FindItemInListBox = -1
End Function

Public Function FindItemInComboBox(cboComboBox As ComboBox, _
                                   queryElem As String) As Integer
  Dim i As Integer
  For i = 0 To cboComboBox.ListCount - 1
    If StrComp(queryElem, cboComboBox.List(i)) = 0 Then
      FindItemInComboBox = i
      Exit Function
    End If
  Next i
  FindItemInComboBox = -1
End Function
```

CLASS - clsCrtCulv (clsCrtCulvTask.cls)

```
Option Explicit

Implements ICommand
Implements ITool

Private m_pRoadLayer As IFeatureLayer
Private m_pCulvLayer As IFeatureLayer
Private m_pSedModel As ISedimentModel

Private m_pApp As IApplication
Private m_pBitmap As IPictureDisp
Private m_pMouseCur As IPictureDisp
Private m_pExt As clsExt
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea

Private m_pDisplay As IDisplay
Private m_pSymbol As ISymbol
Private m_pNewPoint As IPoint
Private m_pSnapAgent As IFeatureSnapAgent

Private m_lRoadFieldInd() As Long
Private m_lCulFieldInd() As Long
' Variables used by the Error handler function - DO NOT REMOVE
Const c_ModuleFileName =
"C:\Evenflo\ThesisWorks\VBScripts\CrossDrainSpacer\clsCreateCulvertTask.
cls"


Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

  'load the button image from the resource file
  Set m_pBitmap = LoadResPicture("Add", vbResBitmap)
  Set m_pMouseCur = LoadResPicture("Digitize", vbResCursor)
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

  Set m_pBitmap = Nothing
  Set m_pMouseCur = Nothing
  Set m_pExt = Nothing
  Set m_pApp = Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  On Error GoTo ErrorHandler

  ICommand_Bitmap = m_pBitmap


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ICommand_Caption = "AddCulv"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ICommand_Category = "Road Sediment Analyst"


  Exit Property
ErrorHandler:
```

```
  HandleError True, "ICommand_Category " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  'check for certain properties in extension
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted And m_pExt.IsAnalyzed Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Property
ErrorHandler:
```

```
  HandleError True, "ICommand_HelpFile " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ICommand_Message = "Add A Culvert And Calculate It's Sediment"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ICommand_Name = "AddCulv"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler


  Dim pMxDoc As IMxDocument
  Set pMxDoc = m_pApp.Document
  Set m_pDisplay = pMxDoc.ActiveView.ScreenDisplay
  Set m_pRoadLayer = m_pExt.RoadLayer
  Set m_pCulvLayer = m_pExt.CulvLayer
  'get all needed field indexes
  m_lRoadFieldInd = Util.GetFieldIndexes(m_pRoadLayer.FeatureClass,
"road")
  m_lCulFieldInd = Util.GetFieldIndexes(m_pCulvLayer.FeatureClass,
"culvert")
  'get the sediment modeler
  Set m_pSedModel = m_pApp.FindExtensionByName(m_pExt.SedModelName)
  If m_pSedModel Is Nothing Then
    MsgBox "Could not find the specified sediment modeler!" & vbLf &
"Please check extensions."
    Exit Sub
  End If
  m_pSedModel.DistanceToStream = m_pExt.DistToStreamsLayer
  m_pSedModel.MaxDeliveryDistance = m_pExt.MaxDeliveryDistance
  'create new snap agent
  Set m_pSnapAgent = New FeatureSnap
  With m_pSnapAgent
    Set .FeatureClass = m_pRoadLayer.FeatureClass
    .HitType = esriGeometryPartBoundary
```

```
  End With
  'create new symbol
  Set m_pSymbol = New SimpleMarkerSymbol
  m_pSymbol.ROP2 = esriROPNotXOrPen
  Dim pMarkSym As IMarkerSymbol
  Set pMarkSym = m_pSymbol 'QI
  Dim myColor As IColor
  Set myColor = New RgbColor
  myColor.RGB = RGB(0, 0, 0)
  pMarkSym.Color = myColor
  pMarkSym.Size = 8
  'get the workspace to edit
  Dim pDS As IDataset
  Set pDS = m_pRoadLayer.FeatureClass
  Set m_pWksEdit = pDS.Workspace
  'start editing
  m_pWksEdit.StartEditing True
  'create new screen refresh
  Set m_pRefresh = New InvalidArea
  Set m_pRefresh.Display = m_pDisplay



  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ICommand_Tooltip = "Add A Culvert And Compute It's Sediment"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
```

```
Private Property Get ITool_Cursor() As esriCore.OLE_HANDLE
  On Error GoTo ErrorHandler

  ITool_Cursor = m_pMouseCur


  Exit Property
ErrorHandler:
  HandleError True, "ITool_Cursor " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Function ITool_Deactivate() As Boolean
  On Error GoTo ErrorHandler

  DrawSymbol m_pNewPoint
  Set m_pNewPoint = Nothing 'this will avoid marker leftovers
  Set m_pDisplay = Nothing
  Set m_pSymbol = Nothing

  If m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StopEditing True
  End If
  Set m_pWksEdit = Nothing
  Set m_pRefresh = Nothing
  Set m_pRoadLayer = Nothing
  Set m_pCulvLayer = Nothing
  Set m_pSedModel = Nothing
  Set m_pSnapAgent = Nothing

  ITool_Deactivate = True


  Exit Function
ErrorHandler:
  HandleError True, "ITool_Deactivate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long)
As Boolean
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Function
ErrorHandler:
  HandleError True, "ITool_OnContextMenu " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Sub ITool_OnDblClick()
  On Error GoTo ErrorHandler

  'unused
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnDblClick " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnKeyDown " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnKeyUp " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler



 Dim pRoadFeature As IFeature
 Dim pPolyline As IPolyline
 Dim pUpperLower(1) As IPolyline
 Dim pLowerSegment As IFeature
 Dim pNewCulvert As IFeature

 If Not m_pNewPoint Is Nothing Then
'    'start editing
'    m_pWksEdit.StartEditing True

    Set pRoadFeature = Util.FindFeatureXPoint(m_pRoadLayer.FeatureClass,
m_pNewPoint)
    'get the feature intersected by the point
    If Not pRoadFeature Is Nothing Then

      Dim lCulIdx As Long, lP1x As Long, lP2x As Long, lP3x As Long,
lRemx As Long, _
```

```
      lCux As Long, lChx As Long, lIdx As Long, lTpx As Long, lFpx As
Long, _
      lDelPotx As Long, lCulSedx As Long, lSedProdx As Long
      lIdx = m_lRoadFieldInd(0)
      lFpx = m_lRoadFieldInd(3)
      lTpx = m_lRoadFieldInd(4)
      lP1x = m_lRoadFieldInd(5)
      lP2x = m_lRoadFieldInd(6)
      lP3x = m_lRoadFieldInd(7)
      lChx = m_lRoadFieldInd(8)
      lCux = m_lRoadFieldInd(9)
      lSedProdx = m_lRoadFieldInd(10)
      lCulIdx = m_lCulFieldInd(0)
      lRemx = m_lCulFieldInd(1)
      lDelPotx = m_lCulFieldInd(2)
      lCulSedx = m_lCulFieldInd(3)


      m_pWksEdit.StartEditOperation
      'remove code is essential in the Remove procedure
      Dim iRemCode As Integer
      iRemCode = 2

      Set pPolyline = pRoadFeature.Shape
      'the max value can be negative. should check for it. see
GetMaxValue
      Dim lNewCul As Long
      lNewCul = Util.GetMaxValue(m_pCulvLayer.FeatureClass, lCulIdx) + 1
      'test to see if point has been placed at upper end, lower end
      'or somewhere else on the line and branch execution accordingly
      Util.CutPolylineAtPoint pPolyline, m_pNewPoint, pUpperLower
      If pUpperLower(0).Length = 0 Then
        'point has been placed at the upper end of a segment
        If Not ConnectFnct.HasParents(pRoadFeature, lP1x, lP2x, lP3x)
Then
          'a culvert must already be present as inseted at set-up
          MsgBox "debug message: can't add there"
          Exit Sub
        End If
      ElseIf pUpperLower(1).Length = 0 Then
        'point has been placed at the lower end of the segment
        Dim lChValue As Long
        lChValue = pRoadFeature.Value(lChx)
        If lChValue = -1 Then
          MsgBox "debug message: can't add on top of another culvert"
          Exit Sub
        Else
          'make road feature point to the child
          Set pRoadFeature =
Util.FindOneFeature(m_pRoadLayer.FeatureClass, _
                                                "RSAID", lChValue)
          'the point is now at the upper end so we apply the above
procedure
        End If
      Else
        'MsgBox "check 1"
        'point has been placed somewhere else on the line
        'set remove code to 1
        iRemCode = 1
```

```
        'split old road in two segments
        'add lower segment as new road feature
        Set pLowerSegment = m_pRoadLayer.FeatureClass.CreateFeature
        Set pLowerSegment.Shape = pUpperLower(1)
        'copy all non original attributes from initial segment
        Util.CopyAllNonRSAtributes pRoadFeature, pLowerSegment,
m_lRoadFieldInd, m_pRoadLayer.FeatureClass.ShapeFieldName
        'give new feat id
        Dim lMaxRoadId As Long
        lMaxRoadId = Util.GetMaxValue(m_pRoadLayer.FeatureClass, lIdx) +
1
        pLowerSegment.Value(lIdx) = lMaxRoadId
        'MsgBox "check 2"
        'set new feature's child and culvert to old roadfeature's values
        Dim lNewPt As Long
        lNewPt = Util.GetMaxOfFields(m_pRoadLayer.FeatureClass, lFpx,
lTpx) + 1
        pLowerSegment.Value(lChx) = pRoadFeature.Value(lChx)
        pLowerSegment.Value(lCux) = pRoadFeature.Value(lCux)
        pLowerSegment.Value(lP1x) = pRoadFeature.Value(lIdx) 'key for
propagation in set upstream
        pLowerSegment.Value(lTpx) = pRoadFeature.Value(lTpx)
        pLowerSegment.Value(lFpx) = lNewPt
        'MsgBox "check 3"
        ConnectFnct.SplitAttrib pLowerSegment, pRoadFeature, lSedProdx
        pLowerSegment.Store
        m_pRefresh.Add pLowerSegment
        'find original child and set one parent to NewF's id
        'MsgBox "check 4"
        ConnectFnct.ChangeOneParent m_pRoadLayer,
pRoadFeature.Value(lChx), pRoadFeature.Value(lIdx), _
                    lMaxRoadId, lP1x, lP2x, lP3x, lIdx

        'change old road shape to upper segment
        Set pRoadFeature.Shape = pUpperLower(0)
        pRoadFeature.Value(lTpx) = lNewPt
        pRoadFeature.Store
        'MsgBox "check 5"
        m_pRefresh.Add pRoadFeature
        'Point road feature to newly added lower segment
        Set pRoadFeature = pLowerSegment
      End If

      'this is needed in any of the 3 cases. Enforces the right flow
      'and adds the culvert to the culvert layer

      'set all dependents upstream to new culvert number
      ConnectFnct.SetUpstream lNewCul, m_pRoadLayer, pRoadFeature, lP1x,
lP2x, lP3x, lCux
      'set parents of this segment to no child and set segment to have
no parents
      ConnectFnct.SevereChild pRoadFeature, m_pRoadLayer, lP1x, lP2x,
lP3x, lChx
      'add culvert to culvert layer
      Set pNewCulvert = m_pCulvLayer.FeatureClass.CreateFeature
      Set pNewCulvert.Shape = m_pNewPoint
      pNewCulvert.Value(lCulIdx) = lNewCul
      pNewCulvert.Value(lRemx) = iRemCode
```

```
      'compute sediment for culvert
      Dim dDelPot As Double
      dDelPot = m_pSedModel.GetDeliveryPotential(m_pNewPoint)
      pNewCulvert.Value(lDelPotx) = dDelPot
      pNewCulvert.Value(lCulSedx) = dDelPot *
ConnectFnct.SumUpSed(m_pRoadLayer.FeatureClass, _
                                    lSedProdx, "CULV", lNewCul)
      pNewCulvert.Store 'store now; new culvert will change
      m_pRefresh.Add m_pNewPoint

      'find the next culvert down and recompute it's sediment
      Set pNewCulvert = Util.FindOneFeature(m_pCulvLayer.FeatureClass,
"RSAID", pRoadFeature.Value(lCux))
      If Not pNewCulvert Is Nothing Then
        pNewCulvert.Value(lCulSedx) = pNewCulvert.Value(lDelPotx) * _

ConnectFnct.SumUpSed(m_pRoadLayer.FeatureClass, _
                                      lSedProdx, "CULV",
pNewCulvert.Value(lCulIdx))
      End If
      pNewCulvert.Store
      m_pRefresh.Add pNewCulvert

      'MsgBox "check 6"
      m_pWksEdit.StopEditOperation
      m_pRefresh.Invalidate esriAllScreenCaches

      'display sediment
      m_pExt.ShowTotalSed
(Util.SumValuesOnField(m_pCulvLayer.FeatureClass, lCulSedx))
    End If
'    m_pWksEdit.StopEditing True
'    MsgBox "saved to database"
  End If

  'release memory
  Set pRoadFeature = Nothing
  Set pPolyline = Nothing
  Set pUpperLower(0) = Nothing
  Set pUpperLower(1) = Nothing
  Set pLowerSegment = Nothing
  Set pNewCulvert = Nothing



  Exit Sub
ErrorHandler:
    m_pWksEdit.StopEditOperation
  m_pWksEdit.UndoEditOperation
'  If m_pWksEdit.IsBeingEdited Then
'    m_pWksEdit.StopEditing False
'  End If
  HandleError True, "ITool_OnMouseDown " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
```

```
  On Error GoTo ErrorHandler

  If Button = 0 Then
    DrawSymbol m_pNewPoint
    Set m_pNewPoint = m_pDisplay.DisplayTransformation.ToMapPoint(X, Y)
    DrawSymbol m_pNewPoint
  End If


  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseMove " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

  'not used


  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseUp " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_Refresh(ByVal hDC As esriCore.OLE_HANDLE)
  On Error GoTo ErrorHandler

  'avoid a marker left on the line
  Set m_pNewPoint = Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "ITool_Refresh " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Sub DrawSymbol(pPoint)
  On Error GoTo ErrorHandler


  If Not pPoint Is Nothing Then 'the point is initialy nothing
    m_pDisplay.StartDrawing m_pDisplay.hDC, esriNoScreenCache
    m_pSymbol.SetupDC m_pDisplay.hDC, m_pDisplay.DisplayTransformation
    m_pSnapAgent.Snap Nothing, pPoint, 100
    m_pSymbol.Draw pPoint
    m_pSymbol.ResetDC
    m_pDisplay.FinishDrawing
  End If


  Exit Sub
```

```
ErrorHandler:
  HandleError True, "DrawSymbol " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub
```

CLASS - clsEnforCon (clsEnforCon.cls)

```
Option Explicit

Implements ICommand

Private m_pApp As IApplication
Private m_pExt As clsExt
Private m_pBitmap As IPictureDisp

Private Sub Class_Initialize()
  Set m_pBitmap = LoadResPicture("Connect", vbResBitmap)
End Sub

Private Sub Class_Terminate()
  Set m_pBitmap = Nothing
  Set m_pApp = Nothing
  Set m_pExt = Nothing
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  ICommand_Bitmap = m_pBitmap
End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "EnforCon"
End Property

Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean
  'TODO: your implementation here
End Property

Private Property Get ICommand_Enabled() As Boolean
   'check for certain properties in extension
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted And m_pExt.IsSetUp And Not m_pExt.HasTopology
Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long
  'TODO: your implementation here
```

```
End Property

Private Property Get ICommand_HelpFile() As String
  'TODO: your implemetation here
End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Enforce Required Ditch Connectivity"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "EnforCon"
End Property

Private Sub ICommand_OnClick()
  Dim pWksEdit As IWorkspaceEdit
  Dim pFeatClass As IFeatureClass
  Dim pMouseCur As IMouseCursor

  On Error GoTo erh
  'get the layer form the extension
  Set pFeatClass = m_pExt.RoadLayer.FeatureClass
  'get the workspace to edit
  Dim pDS As IDataset
  Set pDS = pFeatClass
  Set pWksEdit = pDS.Workspace
  'start editing
  If Not pWksEdit.IsBeingEdited Then
    pWksEdit.StartEditing False
  End If
  'set cursor to busy
  Set pMouseCur = New MouseCursor
  pMouseCur.SetCursor 2
  'enforce simple paths
  EnforceFnct.SimplifyPaths pFeatClass
  'enforce end connectivity only, no midway intersections
  EnforceFnct.ForceEndConnectivity pFeatClass
  'reset cursor
  pMouseCur.SetCursor 0
  'save changes
  pWksEdit.StopEditing True
  'refresh screen
  Dim pDoc As IMxDocument
  Set pDoc = m_pApp.Document
  pDoc.ActiveView.PartialRefresh esriViewGeography, m_pExt.RoadLayer,
Nothing

  'release memory
  Set pDoc = Nothing
  Set pDS = Nothing
  Set pWksEdit = Nothing
  Set pFeatClass = Nothing
  Set pMouseCur = Nothing
  Exit Sub
erh:
  If Not pWksEdit Is Nothing Then
    pWksEdit.StopEditing False
  End If
  If Not pMouseCur Is Nothing Then
```

```
    pMouseCur.SetCursor 0
  End If
  MsgBox "error in ConEnforce " & Error
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String
  ICommand_Tooltip = "Enforce the proper connectivity for ditch
segments"
End Property
```

CLASS - clsExt (clsExtension.cls)

```
Option Explicit

Implements IExtension
Implements IExtensionConfig
Implements IPersistVariant

Private m_pApp As IApplication
Private m_pRoadLayer As IFeatureLayer
Private m_pCulvLayer As IFeatureLayer
Private m_pStreamLayer As IFeatureLayer
Private m_pDEMLayer As IRasterLayer
Private m_pDTSLayer As IRasterLayer

Private m_iMaxDelDist As Integer

Private m_ExtensionState As esriCore.esriExtensionState
Private m_bStarted As Boolean
Private m_bIsSetUp As Boolean
Private m_bHasTopology As Boolean
Private m_bIsAnalyzed As Boolean
Private m_sSedModelName As String
Private m_sGradeFieldName As String

Private m_iDefaultRoadAge As Integer
Private m_sDefaultRoadSurface As String
Private m_sDefaultRoadTraffic As String
Private m_iDefaultRoadGrade As Integer
Private m_iDefaultRoadWidth As Integer
Private m_iDefaultSlopeCover As Integer

Public Event IsStopping()
Public Event ShowSediment(amount As Double)
Private WithEvents m_pActiveViewEvents As Map


Private Sub Class_Initialize()
```

```
  m_ExtensionState = esriESDisabled
  m_bStarted = False
  m_bIsSetUp = False
  m_bHasTopology = False
  m_bIsAnalyzed = False
  m_sSedModelName = "RSA Sediment Modeler"
  m_sGradeFieldName = "grade"

  m_iDefaultRoadAge = 0
  m_sDefaultRoadSurface = "G2"
  m_sDefaultRoadTraffic = "L"
  m_iDefaultRoadWidth = 12
  m_iDefaultRoadGrade = 6
  m_iDefaultSlopeCover = 50

End Sub

Private Property Get IExtension_Name() As String

  IExtension_Name = "Road Sediment Analyst"


End Property

Private Sub IExtension_Shutdown()

  Set m_pCulvLayer = Nothing
  Set m_pRoadLayer = Nothing
  Set m_pApp = Nothing
  Set m_pActiveViewEvents = Nothing


End Sub

Private Sub IExtension_Startup(initializationData As Variant)

  Set m_pApp = initializationData
  Dim pMxDoc As IMxDocument
  Set pMxDoc = m_pApp.Document
  Set m_pActiveViewEvents = pMxDoc.FocusMap


End Sub

Private Property Get IExtensionConfig_Description() As String

  IExtensionConfig_Description = "Relief Culvert Sediment Analysis Tool"


End Property

Private Property Get IExtensionConfig_ProductName() As String

  IExtensionConfig_ProductName = "Road Sediment Analyst"


End Property
```

```
Private Property Let IExtensionConfig_State(ByVal RHS As
esriCore.esriExtensionState)

  m_ExtensionState = RHS


End Property

Private Property Get IExtensionConfig_State() As
esriCore.esriExtensionState

  IExtensionConfig_State = m_ExtensionState


End Property

Private Property Get IPersistVariant_ID() As esriCore.IUID

  IPersistVariant_ID.Value = "RSA option values"


End Property

Private Sub IPersistVariant_Load(ByVal Stream As
esriCore.IVariantStream)
  On Error GoTo erh
 'read data in and set variables accordingly
  m_sSedModelName = Stream.Read
  m_iDefaultRoadAge = Stream.Read
  m_iDefaultRoadGrade = Stream.Read
  m_sDefaultRoadSurface = Stream.Read
  m_iDefaultRoadWidth = Stream.Read
  m_sDefaultRoadTraffic = Stream.Read
  m_iDefaultSlopeCover = Stream.Read
  m_bStarted = Stream.Read
  If m_bStarted Then 'continue reading
    m_bIsSetUp = Stream.Read
    m_bHasTopology = Stream.Read
    m_bIsAnalyzed = Stream.Read
    m_iMaxDelDist = Stream.Read
    Dim pDoc As IMxDocument
    Set pDoc = m_pApp.Document
    Dim bFoundAll As Boolean
    bFoundAll = False
    bFoundAll = Util.ExistsLayer(Stream.Read, m_pRoadLayer, pDoc)
    bFoundAll = Util.ExistsLayer(Stream.Read, m_pCulvLayer, pDoc)
    bFoundAll = Util.ExistsLayer(Stream.Read, m_pDTSLayer, pDoc)
    bFoundAll = Util.ExistsLayer(Stream.Read, m_pStreamLayer, pDoc)
    bFoundAll = Util.ExistsLayer(Stream.Read, m_pDEMLayer, pDoc)
    m_bStarted = bFoundAll
  End If
  'MsgBox "read values : " & m_bStarted & vbLf & m_bIsSetUp & vbLf _
  '                      & m_bHasTopology & vbLf & m_bIsAnalyzed & vbLf
& m_iMaxDelDist
  'display sediment
  If m_bIsAnalyzed Then
    'display sediment when started
    If Not m_pCulvLayer Is Nothing Then
      'find field
```

```
      Dim index As Long
      index = m_pCulvLayer.FeatureClass.FindField("SED")
      If index > -1 Then
        RaiseEvent
ShowSediment(Util.SumValuesOnField(m_pCulvLayer.FeatureClass, index))
      End If
    End If
  End If
  Exit Sub
erh:
  MsgBox "Error ecountered loading variables" & vbLf & Error & vbLf &
"Road Sediment Analyst will reinitialize!"
  m_bStarted = False
  m_bIsSetUp = False
  m_bHasTopology = False
  m_bIsAnalyzed = False
End Sub

Private Sub IPersistVariant_Save(ByVal Stream As
esriCore.IVariantStream)

  Stream.Write m_sSedModelName
  Stream.Write m_iDefaultRoadAge
  Stream.Write m_iDefaultRoadGrade
  Stream.Write m_sDefaultRoadSurface
  Stream.Write m_iDefaultRoadWidth
  Stream.Write m_sDefaultRoadTraffic
  Stream.Write m_iDefaultSlopeCover
  Stream.Write m_bStarted
  Stream.Write m_bIsSetUp
  Stream.Write m_bHasTopology
  Stream.Write m_bIsAnalyzed
  Stream.Write m_iMaxDelDist
  Stream.Write Util.VerifyName(m_pRoadLayer)
  Stream.Write Util.VerifyName(m_pCulvLayer)
  Stream.Write Util.VerifyName(m_pDTSLayer)
  Stream.Write Util.VerifyName(m_pStreamLayer)
  Stream.Write Util.VerifyName(m_pDEMLayer)


End Sub

Public Property Get RoadLayer() As IFeatureLayer

  Set RoadLayer = m_pRoadLayer


End Property

Public Property Let RoadLayer(ByVal RoadLayer As IFeatureLayer)

  Set m_pRoadLayer = RoadLayer
  'DisplayNameControl "Road Layer", m_pRoadLayer


End Property

Public Property Get CulvLayer() As IFeatureLayer
```

```
  Set CulvLayer = m_pCulvLayer

End Property

Public Property Let CulvLayer(ByVal CulvLayer As IFeatureLayer)

  Set m_pCulvLayer = CulvLayer
  'DisplayNameControl "Culvert Layer", m_pCulvLayer

End Property

Public Property Get IsStarted() As Boolean

  IsStarted = m_bStarted

End Property

Public Property Let IsStarted(ByVal Started As Boolean)

  m_bStarted = Started

End Property

Public Property Get StreamLayer() As IFeatureLayer

  Set StreamLayer = m_pStreamLayer

End Property

Public Property Let StreamLayer(ByVal StreamLayer As IFeatureLayer)

  Set m_pStreamLayer = StreamLayer
  'DisplayNameControl "Stream Layer", m_pStreamLayer

End Property

Public Property Get DemLayer() As IRasterLayer

  Set DemLayer = m_pDEMLayer

End Property

Public Property Let DemLayer(ByVal DemLayer As IRasterLayer)

  Set m_pDEMLayer = DemLayer
  'DisplayNameControl "DEM Layer", m_pDEMLayer

End Property

Private Sub DisplayNameControl(sName As String, pData As Variant)

  If pData Is Nothing Then
    MsgBox sName & " set to nothing"
```

```
  ElseIf TypeOf pData Is IDataset Then
    Dim pLayer As IDataset
    Set pLayer = pData
    MsgBox sName & " set to " & pLayer.Name
  Else: MsgBox sName & " set to " & pData
  End If


End Sub

Public Property Get SedModelName() As String

  SedModelName = m_sSedModelName


End Property

Public Property Let SedModelName(ByVal sNewValue As String)

  m_sSedModelName = sNewValue
  'DisplayNameControl "Sediment modeler", sNewValue


End Property

Public Property Get DistToStreamsLayer() As IRasterLayer

  Set DistToStreamsLayer = m_pDTSLayer


End Property

Public Property Let DistToStreamsLayer(ByVal pNewValue As IRasterLayer)

  Set m_pDTSLayer = pNewValue


End Property

Private Sub m_pActiveViewEvents_ItemDeleted(ByVal Item As Variant)

  'if one of the RSA layers is removed
  'will set RSA started to false in order to disable appropriate tools
  If m_bStarted Then
    Dim pLayer As ILayer
    Set pLayer = Item
    'these layers are necessary for both setup and non setup case
    If pLayer Is m_pRoadLayer Or pLayer Is m_pCulvLayer Or _
                                       pLayer Is m_pDTSLayer Then
       m_bIsSetUp = False
       m_bStarted = False
       m_bHasTopology = False
       m_bIsAnalyzed = False
       Exit Sub
    End If
    'when RSA has not been set up the next layers are also necessary
    If Not m_bHasTopology Then
      If pLayer Is m_pStreamLayer Then
        m_bStarted = False
```

```
      End If
    End If
    Set pLayer = Nothing
  End If

End Sub

Public Property Get IsSetUp() As Boolean

  IsSetUp = m_bIsSetUp

End Property

Public Property Let IsSetUp(ByVal bNewValue As Boolean)

  m_bIsSetUp = bNewValue

End Property

Public Property Get HasTopology() As Boolean

  HasTopology = m_bHasTopology

End Property

Public Property Let HasTopology(ByVal bNewValue As Boolean)

  m_bHasTopology = bNewValue

End Property

Public Property Get GradeName() As String

  GradeName = m_sGradeFieldName

End Property

Public Property Let GradeName(ByVal sNewName As String)

  m_sGradeFieldName = sNewName

End Property

Public Property Get IsAnalyzed() As Boolean

  IsAnalyzed = m_bIsAnalyzed

End Property

Public Property Let IsAnalyzed(ByVal bNewValue As Boolean)
```

```
  m_bIsAnalyzed = bNewValue

End Property

Public Property Get MaxDeliveryDistance() As Integer

  MaxDeliveryDistance = m_iMaxDelDist

End Property

Public Property Let MaxDeliveryDistance(ByVal vNewValue As Integer)

  m_iMaxDelDist = vNewValue

End Property

Public Function TriggerStopEvent()

  RaiseEvent IsStopping

End Function

Public Function ShowTotalSed(amount As Double)

  RaiseEvent ShowSediment(amount)

End Function

Public Property Get DefaultRoadAge() As Integer
  DefaultRoadAge = m_iDefaultRoadAge
End Property

Public Property Let DefaultRoadAge(ByVal vNewValue As Integer)
  m_iDefaultRoadAge = vNewValue
End Property

Public Property Get DefaultRoadWidth() As Integer
  DefaultRoadWidth = m_iDefaultRoadWidth
End Property

Public Property Let DefaultRoadWidth(ByVal vNewValue As Integer)
  m_iDefaultRoadWidth = vNewValue
End Property

Public Property Get DefaultRoadGrade() As Integer
  DefaultRoadGrade = m_iDefaultRoadGrade
End Property

Public Property Let DefaultRoadGrade(ByVal vNewValue As Integer)
  m_iDefaultRoadGrade = vNewValue
End Property

Public Property Get DefaultRoadTraffic() As String
  DefaultRoadTraffic = m_sDefaultRoadTraffic
```

```
End Property

Public Property Let DefaultRoadTraffic(ByVal vNewValue As String)
  m_sDefaultRoadTraffic = vNewValue
End Property

Public Property Get DefaultRoadSurface() As String
  DefaultRoadSurface = m_sDefaultRoadSurface
End Property

Public Property Let DefaultRoadSurface(ByVal vNewValue As String)
  m_sDefaultRoadSurface = vNewValue
End Property

Public Property Get DefaultSlopeCover() As Integer
  DefaultSlopeCover = m_iDefaultSlopeCover
End Property

Public Property Let DefaultSlopeCover(ByVal vNewValue As Integer)
  m_iDefaultSlopeCover = vNewValue
End Property
```

CLASS - clsFlip (clsFlip.cls)

```
Option Explicit

Implements ICommand
Implements ITool

Private m_pApp As IApplication
Private m_pBitmap As IPictureDisp
Private m_pExt As clsExt
Private m_pFeatClass As IFeatureClass
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea
Private m_pNewPoint As IPoint
Private m_pDisplay As IDisplay

Private Sub Class_Initialize()
  'load the button image from the resource file
  Set m_pBitmap = LoadResPicture("Flip", vbResBitmap)
End Sub

Private Sub Class_Terminate()
  Set m_pBitmap = Nothing
  Set m_pExt = Nothing
  Set m_pApp = Nothing
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  ICommand_Bitmap = m_pBitmap
End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "Flip"
End Property
```

```
Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean
  'TODO: your implementation here
End Property

Private Property Get ICommand_Enabled() As Boolean
  'check for certain properties in extension
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted And m_pExt.IsSetUp And Not m_pExt.HasTopology
Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long
  'TODO: your implementation here
End Property

Private Property Get ICommand_HelpFile() As String
  'TODO: your implementation here
End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Flip Flow Direction Along Road Ditch"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "Flip"
End Property

Private Sub ICommand_OnClick()
  On Error GoTo erh
  Dim pMxDoc As IMxDocument
  Set pMxDoc = m_pApp.Document
  Set m_pDisplay = pMxDoc.ActiveView.ScreenDisplay
  Set m_pFeatClass = m_pExt.RoadLayer.FeatureClass
  'get the workspace to edit
  Dim pDS As IDataset
  Set pDS = m_pFeatClass
  Set m_pWksEdit = pDS.Workspace
  If Not m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StartEditing True
  End If
  'create new screen refresh
  Set m_pRefresh = New InvalidArea
  Set m_pRefresh.Display = m_pDisplay

  Exit Sub
erh:
  MsgBox "error in create flip cmd: " & Error
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
```

```
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String
  ICommand_Tooltip = "Flip Flow Direction Along Road Ditch"
End Property

Private Property Get ITool_Cursor() As esriCore.OLE_HANDLE

End Property

Private Function ITool_Deactivate() As Boolean
  Set m_pDisplay = Nothing
  Set m_pFeatClass = Nothing
  Set m_pNewPoint = Nothing

  If Not m_pWksEdit Is Nothing Then
    m_pWksEdit.StopEditing True
  End If
  Set m_pWksEdit = Nothing
  Set m_pRefresh = Nothing

  ITool_Deactivate = True
End Function

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long)
As Boolean
  'TODO: your implementation here
End Function

Private Sub ITool_OnDblClick()
  'unused
End Sub

Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Button = 1 Then
    Set m_pNewPoint = m_pDisplay.DisplayTransformation.ToMapPoint(X, Y)
    If Not m_pNewPoint Is Nothing Then
      Dim pFeat As IFeature
      Set pFeat = Util.FindFeatureNearPoint(m_pFeatClass, m_pNewPoint,
10)
      'get the feature intersected by the point
      If Not pFeat Is Nothing Then
        m_pWksEdit.StartEditOperation
        'flip flow direction
        Dim pPolyline As IPolyline
        Set pPolyline = pFeat.Shape
```

```
        pPolyline.ReverseOrientation
        pFeat.Store
        m_pWksEdit.StopEditOperation

        m_pRefresh.Add pFeat
        m_pRefresh.Invalidate esriAllScreenCaches
        'release memory
        Set pFeat = Nothing
      End If
    End If
  End If
End Sub

Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)

End Sub

Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  'not used
End Sub

Private Sub ITool_Refresh(ByVal hDC As esriCore.OLE_HANDLE)

End Sub
```

CLASS - clsMenu (clsMenu.cls)

```
Option Explicit

Implements IMenuDef

Private Property Get IMenuDef_Caption() As String
  IMenuDef_Caption = "Culvert Spacer"
End Property

Private Sub IMenuDef_GetItemInfo(ByVal pos As Long, ByVal itemDef As
esriCore.IItemDef)
  Dim pUID As New UID
  itemDef.Group = False

  Select Case pos
    Case 0
      pUID.Value = "RoadSedimentAnalyst.clsStart"
    Case 1
      pUID.Value = "RoadSedimentAnalyst.clsSetUpRoad"
    Case 2
      pUID.Value = "RoadSedimentAnalyst.clsSetUpFlow"
    Case 3
      pUID.Value = "RoadSedimentAnalyst.clsRunAnalysis"
    Case 4
      pUID.Value = "RoadSedimentAnalyst.clsStop"
    Case 5
      itemDef.Group = True
      pUID.Value = "RoadSedimentAnalyst.clsOptions"
```

```
  End Select

  itemDef.ID = pUID
End Sub

Private Property Get IMenuDef_ItemCount() As Long
  IMenuDef_ItemCount = 6
End Property

Private Property Get IMenuDef_Name() As String
  IMenuDef_Name = "Road Sediment Analyst Menu"
End Property
```

CLASS – clsMerge (clsMerge.cls)

```
Option Explicit

Implements ICommand
Implements ITool

Private m_pApp As IApplication
Private m_pBitmap As IPictureDisp
Private m_pMouseCur As IPictureDisp
Private m_pExt As clsExt
Private m_pFeatClass As IFeatureClass
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea
Private m_pFeatures() As IFeature
Private m_pFeatSel As IFeatureSelection
Dim e1 As Integer, e2 As Integer

Private m_pDisplay As IDisplay
Private m_pSymbol As ISymbol
Private m_pNewPoint As IPoint
Private m_pSnapAgent As IFeatureSnapAgent

Private Sub Class_Initialize()
  'load the button image from the resource file
  Set m_pBitmap = LoadResPicture("Merge", vbResBitmap)
  Set m_pMouseCur = LoadResPicture("EditLine", vbResCursor)
End Sub

Private Sub Class_Terminate()
  Set m_pBitmap = Nothing
  Set m_pMouseCur = Nothing
  Set m_pExt = Nothing
  Set m_pApp = Nothing
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  ICommand_Bitmap = m_pBitmap
End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "Merge"
End Property
```

```
Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean
  'TODO: your implementation here
End Property

Private Property Get ICommand_Enabled() As Boolean
  'check for certain properties in extension
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted And m_pExt.IsSetUp And Not m_pExt.HasTopology
Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long
  'TODO: your implementation here
End Property

Private Property Get ICommand_HelpFile() As String
  'TODO: your implementation here
End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Merge Two Road Segments"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "Merge"
End Property

Private Sub ICommand_OnClick()
  On Error GoTo erh
  Dim pMxDoc As IMxDocument
  Set pMxDoc = m_pApp.Document
  Set m_pDisplay = pMxDoc.ActiveView.ScreenDisplay
  Set m_pFeatClass = m_pExt.RoadLayer.FeatureClass
  'create new snap agent
  Set m_pSnapAgent = New FeatureSnap
  With m_pSnapAgent
    Set .FeatureClass = m_pFeatClass
    .HitType = esriGeometryPartEndpoint
  End With
  'create new symbol
  Set m_pSymbol = New SimpleMarkerSymbol
  m_pSymbol.ROP2 = esriROPNotXOrPen
  Dim pMarkSym As IMarkerSymbol
  Set pMarkSym = m_pSymbol 'QI
  Dim myColor As IColor
  Set myColor = New RgbColor
  myColor.RGB = RGB(0, 0, 0)
  pMarkSym.Color = myColor
  pMarkSym.Size = 8
```

```
  'get the workspace to edit
  Dim pDS As IDataset
  Set pDS = m_pFeatClass
  Set m_pWksEdit = pDS.Workspace
  If Not m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StartEditing True
  End If
  'get feature selection
  Set m_pFeatSel = m_pExt.RoadLayer
  'create new screen refresh
  Set m_pRefresh = New InvalidArea
  Set m_pRefresh.Display = m_pDisplay

  Exit Sub
erh:
  MsgBox "error in create flip cmd: " & Error
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String
  ICommand_Tooltip = "Merge Two Road Segments"
End Property

Private Property Get ITool_Cursor() As esriCore.OLE_HANDLE
  ITool_Cursor = m_pMouseCur
End Property

Private Function ITool_Deactivate() As Boolean
  DrawSymbol m_pNewPoint
  Set m_pNewPoint = Nothing 'this will avoid marker leftovers
  Set m_pDisplay = Nothing
  Set m_pSymbol = Nothing

  m_pFeatSel.Clear

  If Not m_pWksEdit Is Nothing Then
    m_pWksEdit.StopEditing True
  End If
  Set m_pWksEdit = Nothing
  m_pRefresh.Invalidate esriAllScreenCaches
  Set m_pFeatSel = Nothing
  Set m_pRefresh = Nothing

  ITool_Deactivate = True
End Function

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long)
As Boolean
  ITool_OnContextMenu = True
End Function

Private Sub ITool_OnDblClick()
  'unused
```

```
End Sub

Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Button = 1 Then
    If Not m_pNewPoint Is Nothing Then
      m_pFeatures = Util.FindAllFeaturesXPoint(m_pFeatClass,
m_pNewPoint)
      'select two feature from the array
      m_pFeatSel.Clear
      If Not m_pFeatures(0) Is Nothing Then
        If UBound(m_pFeatures) > 0 Then
          SelectTwoFeatures
        End If
      End If
    End If
  Else 'button is 2
    'rotate selection at user's will
    On Error GoTo erh
    If UBound(m_pFeatures) > 1 Then
      SelectTwoFeatures
    End If
  End If

  Exit Sub
erh:
  MsgBox Error
End Sub

Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Button = 0 Then
    DrawSymbol m_pNewPoint
    Set m_pNewPoint = m_pDisplay.DisplayTransformation.ToMapPoint(X, Y)
    DrawSymbol m_pNewPoint
  End If
End Sub

Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Button = 1 Then
    'merge features here
    If e1 <> e2 And (Not m_pFeatures(e1) Is Nothing) And _
                    (Not m_pFeatures(e2) Is Nothing) Then
      m_pWksEdit.StartEditOperation
      'merge features
      Dim pBase As IPolyline
      Set pBase = m_pFeatures(e1).Shape
      Dim pAppend As IPolyline
      Set pAppend = m_pFeatures(e2).Shape
```

```
      Set m_pFeatures(e1).Shape = Util.MergePolylines(pBase, pAppend)
      m_pFeatures(e1).Store
      m_pFeatures(e2).Delete
      m_pRefresh.Add m_pFeatures(e1)

      m_pWksEdit.StopEditOperation
      m_pRefresh.Invalidate esriAllScreenCaches
    End If
    'will reset selected segments when button 1 is pushed
    e1 = 0
    e2 = 0
  End If
End Sub

Private Sub ITool_Refresh(ByVal hDC As esriCore.OLE_HANDLE)
  'avoid a marker left on the line
  Set m_pNewPoint = Nothing
End Sub

Sub DrawSymbol(pPoint)
  On Error GoTo erh
  If Not pPoint Is Nothing Then 'the point is initialy nothing
    m_pDisplay.StartDrawing m_pDisplay.hDC, esriNoScreenCache
    m_pSymbol.SetupDC m_pDisplay.hDC, m_pDisplay.DisplayTransformation
    m_pSnapAgent.Snap Nothing, pPoint, 100
    m_pSymbol.Draw pPoint
    m_pSymbol.ResetDC
    m_pDisplay.FinishDrawing
  End If
  Exit Sub
erh:
  MsgBox "error in draw symbol " & Error
End Sub

Private Sub SelectTwoFeatures()
  On Error GoTo erh
    Dim max As Integer, min As Integer
    max = UBound(m_pFeatures)
    min = LBound(m_pFeatures)

    e2 = e2 + 1
    If e2 > max Then
      e1 = e1 + 1
      If e1 = max Then
        e1 = min
      End If
      e2 = e1 + 1
    End If
    Debug.Print max & " " & min & "  " & e1 & " " & e2

    'rotate selection
    m_pFeatSel.Clear
    m_pFeatSel.Add m_pFeatures(e1)
    m_pFeatSel.Add m_pFeatures(e2)
    m_pRefresh.Add m_pFeatures(e1)
    m_pRefresh.Add m_pFeatures(e2)
    m_pRefresh.Invalidate esriAllScreenCaches
  Exit Sub
erh:
```

```
  MsgBox Error
End Sub
```

CLASS – clsMoveCulv (clsMoveCulv.cls)

```
Option Explicit

Implements ICommand
Implements ITool

Private m_pApp As IApplication
Private m_pBitmap As IPictureDisp
Private m_pMouseCur As IPictureDisp
Private m_pExt As clsExt
Private m_pFeatClass As IFeatureClass
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea
Private m_pFeatures() As IFeature
Private m_pFeatSel As IFeatureSelection
Dim e1 As Integer, e2 As Integer

Private m_pDisplay As IDisplay
Private m_pSymbol As ISymbol
Private m_pNewPoint As IPoint
Private m_pSnapAgent As IFeatureSnapAgent

Private Sub Class_Initialize()
  'load the button image from the resource file
  Set m_pBitmap = LoadResPicture("Merge", vbResBitmap)
  Set m_pMouseCur = LoadResPicture("EditLine", vbResCursor)
End Sub

Private Sub Class_Terminate()
  Set m_pBitmap = Nothing
  Set m_pMouseCur = Nothing
  Set m_pExt = Nothing
  Set m_pApp = Nothing
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  ICommand_Bitmap = m_pBitmap
End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "Merge"
End Property

Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean
  'TODO: your implementation here
End Property

Private Property Get ICommand_Enabled() As Boolean
```

```
  'check for certain properties in extension
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted And m_pExt.IsSetUp And Not m_pExt.HasTopology
Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long
  'TODO: your implementation here
End Property

Private Property Get ICommand_HelpFile() As String
  'TODO: your implementation here
End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Merge Two Road Segments"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "Merge"
End Property

Private Sub ICommand_OnClick()
  On Error GoTo erh
  Dim pMxDoc As IMxDocument
  Set pMxDoc = m_pApp.Document
  Set m_pDisplay = pMxDoc.ActiveView.ScreenDisplay
  Set m_pFeatClass = m_pExt.RoadLayer.FeatureClass
  'create new snap agent
  Set m_pSnapAgent = New FeatureSnap
  With m_pSnapAgent
    Set .FeatureClass = m_pFeatClass
    .HitType = esriGeometryPartEndpoint
  End With
  'create new symbol
  Set m_pSymbol = New SimpleMarkerSymbol
  m_pSymbol.ROP2 = esriROPNotXOrPen
  Dim pMarkSym As IMarkerSymbol
  Set pMarkSym = m_pSymbol 'QI
  Dim myColor As IColor
  Set myColor = New RgbColor
  myColor.RGB = RGB(0, 0, 0)
  pMarkSym.Color = myColor
  pMarkSym.Size = 8
  'get the workspace to edit
  Dim pDS As IDataset
  Set pDS = m_pFeatClass
  Set m_pWksEdit = pDS.Workspace
  If Not m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StartEditing True
  End If
  'get feature selection
  Set m_pFeatSel = m_pExt.RoadLayer
  'create new screen refresh
```

```
  Set m_pRefresh = New InvalidArea
  Set m_pRefresh.Display = m_pDisplay

  Exit Sub
erh:
  MsgBox "error in create flip cmd: " & Error
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String
  ICommand_Tooltip = "Merge Two Road Segments"
End Property

Private Property Get ITool_Cursor() As esriCore.OLE_HANDLE
  ITool_Cursor = m_pMouseCur
End Property

Private Function ITool_Deactivate() As Boolean
  DrawSymbol m_pNewPoint
  Set m_pNewPoint = Nothing 'this will avoid marker leftovers
  Set m_pDisplay = Nothing
  Set m_pSymbol = Nothing

  m_pFeatSel.Clear

  If Not m_pWksEdit Is Nothing Then
    m_pWksEdit.StopEditing True
  End If
  Set m_pWksEdit = Nothing
  m_pRefresh.Invalidate esriAllScreenCaches
  Set m_pFeatSel = Nothing
  Set m_pRefresh = Nothing

  ITool_Deactivate = True
End Function

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long)
As Boolean
  ITool_OnContextMenu = True
End Function

Private Sub ITool_OnDblClick()
  'unused
End Sub

Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub
```

```
Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Button = 1 Then
    If Not m_pNewPoint Is Nothing Then
      m_pFeatures = Util.FindAllFeaturesXPoint(m_pFeatClass,
m_pNewPoint)
      'select two feature from the array
      m_pFeatSel.Clear
      If Not m_pFeatures(0) Is Nothing Then
        If UBound(m_pFeatures) > 0 Then
          SelectTwoFeatures
        End If
      End If
    End If
  Else 'button is 2
    'rotate selection at user's will
    On Error GoTo erh
    If UBound(m_pFeatures) > 1 Then
      SelectTwoFeatures
    End If
  End If

  Exit Sub
erh:
  MsgBox Error
End Sub

Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Button = 0 Then
    DrawSymbol m_pNewPoint
    Set m_pNewPoint = m_pDisplay.DisplayTransformation.ToMapPoint(X, Y)
    DrawSymbol m_pNewPoint
  End If
End Sub

Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Button = 1 Then
    'merge features here
    If e1 <> e2 And (Not m_pFeatures(e1) Is Nothing) And _
                    (Not m_pFeatures(e2) Is Nothing) Then
      m_pWksEdit.StartEditOperation
      'merge features
      Dim pBase As IPolyline
      Set pBase = m_pFeatures(e1).Shape
      Dim pAppend As IPolyline
      Set pAppend = m_pFeatures(e2).Shape

      Set m_pFeatures(e1).Shape = Util.MergePolylines(pBase, pAppend)
      m_pFeatures(e1).Store
      m_pFeatures(e2).Delete
      m_pRefresh.Add m_pFeatures(e1)

      m_pWksEdit.StopEditOperation
      m_pRefresh.Invalidate esriAllScreenCaches
    End If
    'will reset selected segments when button 1 is pushed
    e1 = 0
```

```
    e2 = 0
  End If
End Sub

Private Sub ITool_Refresh(ByVal hDC As esriCore.OLE_HANDLE)
  'avoid a marker left on the line
  Set m_pNewPoint = Nothing
End Sub

Sub DrawSymbol(pPoint)
  On Error GoTo erh
  If Not pPoint Is Nothing Then 'the point is initialy nothing
    m_pDisplay.StartDrawing m_pDisplay.hDC, esriNoScreenCache
    m_pSymbol.SetupDC m_pDisplay.hDC, m_pDisplay.DisplayTransformation
    m_pSnapAgent.Snap Nothing, pPoint, 100
    m_pSymbol.Draw pPoint
    m_pSymbol.ResetDC
    m_pDisplay.FinishDrawing
  End If
  Exit Sub
erh:
  MsgBox "error in draw symbol " & Error
End Sub

Private Sub SelectTwoFeatures()
  On Error GoTo erh
    Dim max As Integer, min As Integer
    max = UBound(m_pFeatures)
    min = LBound(m_pFeatures)

    e2 = e2 + 1
    If e2 > max Then
      e1 = e1 + 1
      If e1 = max Then
        e1 = min
      End If
      e2 = e1 + 1
    End If
    Debug.Print max & " " & min & "  " & e1 & " " & e2

    'rotate selection
    m_pFeatSel.Clear
    m_pFeatSel.Add m_pFeatures(e1)
    m_pFeatSel.Add m_pFeatures(e2)
    m_pRefresh.Add m_pFeatures(e1)
    m_pRefresh.Add m_pFeatures(e2)
    m_pRefresh.Invalidate esriAllScreenCaches
  Exit Sub
erh:
  MsgBox Error
End Sub
```

CLASS - clsNodeGrade (clsNodeGrade.cls)

```
Option Explicit

Implements ICommand
```

```
Implements ITool

Private m_pApp As IApplication
Private m_pBitmap As IPictureDisp
Private m_pMouseCur As IPictureDisp
Private m_pExt As clsExt
Private m_pFeatClass As IFeatureClass
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea
Private m_pFeatures() As IFeature
Private m_lGradeIndex As Long
Private m_bDraw As Boolean

Private m_pDisplay As IDisplay
Private m_pSymbol As ISymbol
Private m_pNewPoint As IPoint
Private m_pSnapAgent As IFeatureSnapAgent

Private Sub Class_Initialize()
  'load the button image from the resource file
  Set m_pBitmap = LoadResPicture("NodeGrade", vbResBitmap)
  Set m_pMouseCur = LoadResPicture("Edit", vbResCursor)
  m_lGradeIndex = -1
  m_bDraw = True
End Sub

Private Sub Class_Terminate()
  Set m_pBitmap = Nothing
  Set m_pMouseCur = Nothing
  Set m_pExt = Nothing
  Set m_pApp = Nothing
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  ICommand_Bitmap = m_pBitmap
End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "NodeGrade"
End Property

Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean
  'TODO: your implementation here
End Property

Private Property Get ICommand_Enabled() As Boolean
  'check for certain properties in extension
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted And m_pExt.IsSetUp And Not m_pExt.HasTopology
Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If
```

```
End Property

Private Property Get ICommand_HelpContextID() As Long
  'TODO: your implementation here
End Property

Private Property Get ICommand_HelpFile() As String
  'TODO: your implementation here
End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Adjust Road Grade"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "NodeGrade"
End Property

Private Sub ICommand_OnClick()
  On Error GoTo erh
  Dim pMxDoc As IMxDocument
  Set pMxDoc = m_pApp.Document
  Set m_pDisplay = pMxDoc.ActiveView.ScreenDisplay
  Set m_pFeatClass = m_pExt.RoadLayer.FeatureClass
  'create new snap agent
  Set m_pSnapAgent = New FeatureSnap
  With m_pSnapAgent
    Set .FeatureClass = m_pFeatClass
    .HitType = esriGeometryPartEndpoint
  End With
  'create new symbol
  Set m_pSymbol = New SimpleMarkerSymbol
  m_pSymbol.ROP2 = esriROPNotXOrPen
  Dim pMarkSym As IMarkerSymbol
  Set pMarkSym = m_pSymbol 'QI
  Dim myColor As IColor
  Set myColor = New RgbColor
  myColor.RGB = RGB(0, 0, 0)
  pMarkSym.Color = myColor
  pMarkSym.Size = 8
  'get the workspace to edit
  Dim pDS As IDataset
  Set pDS = m_pFeatClass
  Set m_pWksEdit = pDS.Workspace
  If Not m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StartEditing True
  End If
  'get grade field index
  m_lGradeIndex = m_pFeatClass.FindField(m_pExt.GradeName)

  'create new screen refresh
  Set m_pRefresh = New InvalidArea
  Set m_pRefresh.Display = m_pDisplay

  Exit Sub
erh:
  MsgBox "error in create flip cmd: " & Error
End Sub
```

```
Private Sub ICommand_OnCreate(ByVal hook As Object)
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String
  ICommand_Tooltip = "Manualy Adjust Road Grade At Nodes"
End Property

Private Property Get ITool_Cursor() As esriCore.OLE_HANDLE
  ITool_Cursor = m_pMouseCur
End Property

Private Function ITool_Deactivate() As Boolean
  DrawSymbol m_pNewPoint
  Set m_pNewPoint = Nothing 'this will avoid marker leftovers
  Set m_pDisplay = Nothing
  Set m_pSymbol = Nothing

  If Not m_pWksEdit Is Nothing Then
    m_pWksEdit.StopEditing True
  End If
  Set m_pWksEdit = Nothing

  ITool_Deactivate = True
End Function

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long)
As Boolean
  ITool_OnContextMenu = True
End Function

Private Sub ITool_OnDblClick()
  'unused
End Sub

Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Not m_pNewPoint Is Nothing Then
    m_pFeatures = Util.FindAllFeaturesXPoint(m_pFeatClass, m_pNewPoint)
    m_pWksEdit.StartEditOperation
    If Button = 1 Then
       'increase grade
       AdjustGrade 1
    Else 'button is 2
       'decrease grade
       AdjustGrade -1
    End If
    m_pWksEdit.StartEditOperation
```

```
  End If
End Sub

Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Button = 0 Then
    DrawSymbol m_pNewPoint
    Set m_pNewPoint = m_pDisplay.DisplayTransformation.ToMapPoint(X, Y)
    m_bDraw = True
    DrawSymbol m_pNewPoint
  End If
End Sub

Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  'not used
End Sub

Private Sub ITool_Refresh(ByVal hDC As esriCore.OLE_HANDLE)
  'avoid a marker left on the line
  m_bDraw = False
End Sub

Sub DrawSymbol(pPoint)
  On Error GoTo erh
  If Not pPoint Is Nothing And m_bDraw Then 'the point is initialy
nothing
    m_pDisplay.StartDrawing m_pDisplay.hDC, esriNoScreenCache
    m_pSymbol.SetupDC m_pDisplay.hDC, m_pDisplay.DisplayTransformation
    m_pSnapAgent.Snap Nothing, pPoint, 100
    m_pSymbol.Draw pPoint
    m_pSymbol.ResetDC
    m_pDisplay.FinishDrawing
  End If
  Exit Sub
erh:
  MsgBox "error in draw symbol " & Error
End Sub

Private Sub AdjustGrade(iStep As Integer)
  If m_lGradeIndex > -1 Then
    Dim i As Integer
    For i = 0 To UBound(m_pFeatures)
      If Not m_pFeatures(i) Is Nothing Then
        m_pFeatures(i).Value(m_lGradeIndex) =
m_pFeatures(i).Value(m_lGradeIndex) + iStep
        m_pFeatures(i).Store
        m_pRefresh.Add m_pFeatures(i).Extent
      End If
    Next i
    m_pRefresh.Invalidate esriAllScreenCaches
  End If
End Sub
```

CLASS - clsOptions (clsOptions.cls)

```
Option Explicit
```

```
Implements ICommand
Private m_pApp As IApplication
Private m_pExtension As clsExt

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  'your implementation here
End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "Options"
End Property

Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean
  'your implementation here
End Property

Private Property Get ICommand_Enabled() As Boolean
  If Not m_pExtension Is Nothing Then
'    If Not m_pExtension.IsAnalyzed Then
'      ICommand_Enabled = True
'    Else: ICommand_Enabled = False
'    End If
    ICommand_Enabled = True
  Else: ICommand_Enabled = False
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long
  'your implementation here
End Property

Private Property Get ICommand_HelpFile() As String
  'your implementation here
End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Sets Analysis Options"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "options"
End Property

Private Sub ICommand_OnClick()
  'show the form and set it up
  Dim pOptFrm As frmOptions
  Set pOptFrm = New frmOptions
  pOptFrm.SetUpBoxes m_pExtension, m_pApp
  pOptFrm.Show vbModal
  'form is modal thread would interupt until form is done
  Set pOptFrm = Nothing
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
```

```
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExtension = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String
  ICommand_Tooltip = "Sets Analysis Options"
End Property
```

CLASS - clsRmvCulv (clsRemoveCulv.cls)

```
Option Explicit

Implements ICommand
Implements ITool

Private m_pApp As IApplication
Private m_pDoc As IMxDocument
Private m_pExt As clsExt
Private m_pRoadLayer As IFeatureLayer
Private m_pCulvLayer As IFeatureLayer

Private m_pBitmap As IPictureDisp
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea

Private m_pDisplay As IDisplay
Private m_pSymbol As ISymbol
Private m_pNewPoint As IPoint
Private m_pSnapAgent As IFeatureSnapAgent
' Variables used by the Error handler function - DO NOT REMOVE
Const c_ModuleFileName =
"C:\Evenflo\ThesisWorks\VBScripts\CrossDrainSpacer\clsRemoveCulvert.cls"


Private Sub Class_Initialize()
  On Error GoTo ErrorHandler

27:   Set m_pBitmap = LoadResPicture("Remove", vbResBitmap)


  Exit Sub
ErrorHandler:
  HandleError True, "Class_Initialize " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

38:   Set m_pBitmap = Nothing
39:   Set m_pExt = Nothing
40:   Set m_pApp = Nothing
```

```
  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  On Error GoTo ErrorHandler

51:   ICommand_Bitmap = m_pBitmap


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

62:   ICommand_Caption = "Remove Culvert"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

73:   ICommand_Category = "Road Sediment Analyst"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
```

```
Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  'check for certain properties in extension
96:   If Not m_pExt Is Nothing Then
97:     If m_pExt.IsStarted And m_pExt.IsAnalyzed Then
98:       ICommand_Enabled = True
99:     Else: ICommand_Enabled = False
100:     End If
101:   Else: ICommand_Enabled = False
102:   End If


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  'TODO: your implemetation here


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

135:   ICommand_Message = "Remove Culvert"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
```

```
Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

146:   ICommand_Name = "Remove Culvert"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler


  Dim pMxDoc As IMxDocument
159:   Set pMxDoc = m_pApp.Document
160:   Set m_pDisplay = pMxDoc.ActiveView.ScreenDisplay
161:   Set m_pRoadLayer = m_pExt.RoadLayer
162:   Set m_pCulvLayer = m_pExt.CulvLayer
  'create new snap agent
164:   Set m_pSnapAgent = New FeatureSnap
165:   With m_pSnapAgent
166:     Set .FeatureClass = m_pCulvLayer.FeatureClass
167:     .HitType = esriGeometryPartBoundary
168:   End With
  'create new symbol
170:   Set m_pSymbol = New SimpleMarkerSymbol
171:   m_pSymbol.ROP2 = esriROPNotXOrPen
  Dim pMarkSym As IMarkerSymbol
173:   Set pMarkSym = m_pSymbol 'QI
  Dim myColor As IColor
175:   Set myColor = New RgbColor
176:   myColor.RGB = RGB(0, 0, 0)
177:   pMarkSym.Color = myColor
178:   pMarkSym.Size = 8
  'get the workspace to edit
  Dim pDS As IDataset
181:   Set pDS = m_pRoadLayer.FeatureClass
182:   Set m_pWksEdit = pDS.Workspace
  'start editing
184:   m_pWksEdit.StartEditing True
  'create new screen refresh
186:   Set m_pRefresh = New InvalidArea
187:   Set m_pRefresh.Display = m_pDisplay

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler
```

```
197:   Set m_pApp = hook
198:   Set m_pDoc = m_pApp.Document
  Dim pId As New UID
200:   pId.Value = "RoadSedimentAnalyst.clsExt"
201:   Set m_pExt = m_pApp.FindExtensionByCLSID(pId)


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

212:   ICommand_Tooltip = "Removes Culverts placed with Create Culvert
Tool"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Sub RecalcSed(lCulId As Long, lSedx As Long, lSedProdx As Long,
_
                      lDelPotx As Long, pRefresh As IInvalidArea)
  On Error GoTo ErrorHandler

  Dim pCul As IFeature
225:   Set pCul = Util.FindOneFeature(m_pCulvLayer.FeatureClass,
"RSAID", lCulId)
226:   If Not pCul Is Nothing Then
227:     pCul.Value(lSedx) =
ConnectFnct.SumUpSed(m_pRoadLayer.FeatureClass, lSedProdx, "CULV", _
                        lCulId) * pCul.Value(lDelPotx)
229:     pCul.Store
230:     pRefresh.Add pCul
231:   End If

  'release memory
234:   Set pCul = Nothing


  Exit Sub
ErrorHandler:
  HandleError False, "RecalcSed " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Property Get ITool_Cursor() As esriCore.OLE_HANDLE
  On Error GoTo ErrorHandler
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ITool_Cursor " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Function ITool_Deactivate() As Boolean
  On Error GoTo ErrorHandler

256:   DrawSymbol m_pNewPoint
257:   Set m_pNewPoint = Nothing 'this will avoid marker leftovers
258:   Set m_pDisplay = Nothing
259:   Set m_pSymbol = Nothing

261:   If m_pWksEdit.IsBeingEdited Then
262:     m_pWksEdit.StopEditing True
263:   End If
264:   Set m_pWksEdit = Nothing
265:   Set m_pRefresh = Nothing
266:   Set m_pRoadLayer = Nothing
267:   Set m_pCulvLayer = Nothing
268:   Set m_pSnapAgent = Nothing

270:   ITool_Deactivate = True


  Exit Function
ErrorHandler:
  HandleError True, "ITool_Deactivate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long)
As Boolean
  On Error GoTo ErrorHandler




  Exit Function
ErrorHandler:
  HandleError True, "ITool_OnContextMenu " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Sub ITool_OnDblClick()
  On Error GoTo ErrorHandler



  Exit Sub
ErrorHandler:
```

```
  HandleError True, "ITool_OnDblClick " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  On Error GoTo ErrorHandler



  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnKeyDown " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  On Error GoTo ErrorHandler



  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnKeyUp " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

  Dim pCulvFeat As IFeature
  Dim pPoint As IPoint
  Dim pSegments() As IFeature
  Dim pTopoOpt As ITopologicalOperator
  Dim pChild As IFeature

331:   If Not m_pNewPoint Is Nothing Then
        'MsgBox "debug: point location is " & m_pNewPoint.X & "  " &
m_pNewPoint.Y
'    'start editing
'    m_pWksEdit.StartEditing True

335:     Set pCulvFeat =
Util.FindFeatureXPoint(m_pCulvLayer.FeatureClass, m_pNewPoint)
    'get the feature intersected by the point
337:     If Not pCulvFeat Is Nothing Then
        'MsgBox "debug: found culvert " & pCulvFeat.OID
      'identify required fields
      Dim lChx As Long, lIdx As Long, lCux As Long, lP1x As Long, lP2x
As Long
      Dim lP3x As Long, lRmx As Long, lTpx As Long, lFpx As Long
      Dim lSedProdx As Long, lDelPotx As Long, lSedx As Long
342:       lChx = m_pRoadLayer.FeatureClass.FindField("CHILD")
343:       lIdx = m_pRoadLayer.FeatureClass.FindField("RSAID")
```

```
344:        lCux = m_pRoadLayer.FeatureClass.FindField("CULV")
345:        lP1x = m_pRoadLayer.FeatureClass.FindField("PAR1")
346:        lP2x = m_pRoadLayer.FeatureClass.FindField("PAR2")
347:        lP3x = m_pRoadLayer.FeatureClass.FindField("PAR3")
348:        lRmx = m_pCulvLayer.FeatureClass.FindField("REMCD")
349:        lTpx = m_pRoadLayer.FeatureClass.FindField("TOPT")
350:        lFpx = m_pRoadLayer.FeatureClass.FindField("FROMPT")
351:        lSedx = m_pCulvLayer.FeatureClass.FindField("SED")
352:        lDelPotx = m_pCulvLayer.FeatureClass.FindField("DELPOT")
353:        lSedProdx = m_pRoadLayer.FeatureClass.FindField("SEDPROD")

      'start operation
356:        m_pWksEdit.StartEditOperation

358:        Set pPoint = pCulvFeat.Shape
      'verify the remove code for this culvert
      Select Case pCulvFeat.Value(lRmx)
        Case 0
          'if remove code is 0 can't remove
363:            MsgBox "debug mesage: cannot remove this culvert"
364:            m_pWksEdit.StopEditing False
          Exit Sub
        Case 1
367:            'MsgBox "check 1"
          'if code is 1 -> most common case, culvert was placed on the
line
          'get cursor into roads that satisfy the intersection filter
370:            pSegments = ConnectFnct.IdentifyUpperLower(m_pRoadLayer,
pPoint, lTpx)
371:          ' MsgBox "check 2"
372:            If Not pSegments(0) Is Nothing And Not pSegments(1) Is
Nothing Then
            'the uppersegment will remain, lower will be deleted
            'set uppersegmnet's attributes as to maintain flow integrity
375:              pSegments(0).Value(lChx) = pSegments(1).Value(lChx)
376:              pSegments(0).Value(lCux) = pSegments(1).Value(lCux)
377:            'MsgBox "check 3"
            'change culvert for all parents of the upper segment
379:              ConnectFnct.SetUpstream pSegments(1).Value(lCux),
m_pRoadLayer, _
                                        pSegments(0), lP1x, lP2x, lP3x, lCux
381:            ' MsgBox "check 4"
            'set child of lower to be have upper as parent
383:              ConnectFnct.ChangeOneParent m_pRoadLayer,
pSegments(1).Value(lChx), _
                                        pSegments(1).Value(lIdx),
pSegments(0).Value(lIdx), _
                                        lP1x, lP2x, lP3x, lIdx
386:            ' MsgBox "check 5"
            'union the two road polylines
388:              Set pTopoOpt = pSegments(0).Shape
389:              Set pSegments(0).Shape =
pTopoOpt.Union(pSegments(1).Shape)
            'set upper to flow to lower's to point
391:              pSegments(0).Value(lTpx) = pSegments(1).Value(lTpx)
            'add sediment values from both and assing to upper
393:              ConnectFnct.SumAttrib pSegments(0), pSegments(1),
lSedProdx
394:              pSegments(0).Store
```

```
395:            '  MsgBox "check 6"
396:              m_pRefresh.Add pSegments(0)
397:            '  MsgBox "check 7"
              'remove the lower road segment
399:              pSegments(1).Delete
400:              pCulvFeat.Delete
              'recalculate sediment for culvert below the one being
removed
402:              RecalcSed pSegments(0).Value(lCux), lSedx, lSedProdx,
lDelPotx, m_pRefresh
403:              m_pRefresh.Invalidate esriAllScreenCaches

              'clean up
406:              Set pSegments(0) = Nothing
407:              Set pSegments(1) = Nothing
408:            End If
          Case 2
            'if remove code is 2 there is no need to merge anything,
            'just rebuild the flow setting parents, child,
            'and culvert for all affected segments and remove culvert from
it's layer
413:            pSegments = ConnectFnct.IdentifyUpperLower(m_pRoadLayer,
pPoint, lTpx)
414:            ConnectFnct.ReuniteChild pSegments, lP1x, lP2x, lP3x,
lChx, lIdx
415:            Set pChild = pSegments(UBound(pSegments))
416:            If Not pChild Is Nothing Then
417:              ConnectFnct.SetUpstream pChild.Value(lCux),
m_pRoadLayer, pChild, _
                                        lP1x, lP2x, lP3x, lCux
419:            End If
            'recalculate sediment for culvert below the one being removed
421:              RecalcSed pChild.Value(lCux), lSedx, lSedProdx, lDelPotx,
m_pRefresh
            'remove culvert
423:              m_pRefresh.Add pCulvFeat.Extent
424:              pCulvFeat.Delete
425:              m_pRefresh.Invalidate esriAllScreenCaches

            'Clean up
            Dim i As Integer
429:              For i = LBound(pSegments) To UBound(pSegments)
430:                Set pSegments(i) = Nothing
431:              Next i
432:         End Select
433:         'MsgBox "check 8"
434:         m_pWksEdit.StopEditOperation

       'display sediment
437:         m_pExt.ShowTotalSed
(Util.SumValuesOnField(m_pCulvLayer.FeatureClass, lSedx))
           Else
              MsgBox "Error: Could Not Find a Culvert at Mouse Location"
438:       End If
'    m_pWksEdit.StopEditing True
'    MsgBox "saved to database"
441:     End If

  'release memory
```

```
444:   Set pCulvFeat = Nothing
445:   Set pTopoOpt = Nothing
446:   Set pPoint = Nothing
447:   Set pChild = Nothing

  Exit Sub
ErrorHandler:
451:   m_pWksEdit.StopEditOperation
452:   m_pWksEdit.UndoEditOperation
  'm_pWksEdit.StopEditing False
  HandleError True, "ITool_OnMouseDown " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

460:   If Button = 0 Then
461:     DrawSymbol m_pNewPoint
462:     Set m_pNewPoint =
m_pDisplay.DisplayTransformation.ToMapPoint(X, Y)
463:     DrawSymbol m_pNewPoint
464:   End If


  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseMove " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  On Error GoTo ErrorHandler

  'not used


  Exit Sub
ErrorHandler:
  HandleError True, "ITool_OnMouseUp " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ITool_Refresh(ByVal hDC As esriCore.OLE_HANDLE)
  On Error GoTo ErrorHandler

   'avoid a marker left on the line
487:   Set m_pNewPoint = Nothing


  Exit Sub
ErrorHandler:
```

```
  HandleError True, "ITool_Refresh " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Sub DrawSymbol(pPoint)
  On Error GoTo ErrorHandler


499:   If Not pPoint Is Nothing Then 'the point is initialy nothing
500:     m_pDisplay.StartDrawing m_pDisplay.hDC, esriNoScreenCache
501:     m_pSymbol.SetupDC m_pDisplay.hDC,
m_pDisplay.DisplayTransformation
502:     m_pSnapAgent.Snap Nothing, pPoint, 100
503:     m_pSymbol.Draw pPoint
504:     m_pSymbol.ResetDC
505:     m_pDisplay.FinishDrawing
506:   End If



  Exit Sub
ErrorHandler:
  HandleError True, "DrawSymbol " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub
```

CLASS – clsRunAnalysis (clsRunAnalysis.cls)

```
Option Explicit

Implements ICommand

Private m_pApp As IApplication
Private m_pDoc As IMxDocument
Private m_pExtension As clsExt
Private m_pRoadLayer As IFeatureLayer
Private m_pCulvertLayer As IFeatureLayer
Private m_pDEMLayer As IRasterLayer
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea
Private m_lRoadFieldInd() As Long
Private m_lCulFieldInd() As Long
Private m_pSedModel As ISedimentModel
' Variables used by the Error handler function - DO NOT REMOVE
Const c_ModuleFileName = "C:\CrossDrainSpacer\clsRunAnalysis.cls"
' Constant reflect file module name

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

   If Not m_pExtension Is Nothing Then
    If m_pExtension.HasTopology Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
```

```
    End If
  Else: ICommand_Enabled = False
  End If

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ICommand_Name = "SedimentAnalyzer"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ICommand_Caption = "Analyze Sed"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ICommand_Tooltip = "Run The Sediment Analysis"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
```

```
Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ICommand_Message = "Runs The Sediment Analysis for Culvert System"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  On Error GoTo ErrorHandler

  ' TODO: Add your implementation here

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ICommand_Category = "Road Sediment Analyst"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
```

```
Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  Set m_pApp = hook
  Set m_pDoc = m_pApp.Document
  Set m_pRefresh = New InvalidArea
  Set m_pRefresh.Display = m_pDoc.ActiveView.ScreenDisplay
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExtension = m_pApp.FindExtensionByCLSID(pId)

  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  'verify that the map units are set
  Dim units As esriUnits
  units = m_pDoc.FocusMap.MapUnits
  If units = esriUnknownUnits Or units = esriDecimalDegrees Or units =
esriPoints Or _
    units = esriNauticalMiles Then
    MsgBox "Map units must be set and/or in a road applicable system" &
vbLf & _
           "Please check map units!"
    Exit Sub
  End If

  'Get the sediment modeler
  Set m_pSedModel =
m_pApp.FindExtensionByName(m_pExtension.SedModelName)

  If m_pSedModel Is Nothing Then
    MsgBox "No Sediment Model could be found!" & vbLf & "Please check
extesions."
    Exit Sub
  End If

'  'disable buttons
'  If m_pExtension.IsAnalyzed = False Then

  'get layers from extension
  Set m_pRoadLayer = m_pExtension.RoadLayer
  Set m_pCulvertLayer = m_pExtension.CulvLayer
  Set m_pDEMLayer = m_pExtension.DemLayer

  Dim success As Boolean

  Dim pWks As IWorkspace
  Dim pDS As IDataset
  Set pDS = m_pRoadLayer
  Set pWks = pDS.Workspace
  Set m_pWksEdit = pWks
```

```
   If m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StopEditing True
  End If

  'get field indexes in array
  m_lRoadFieldInd = Util.GetFieldIndexes(m_pRoadLayer.FeatureClass,
"road")
  'get grade field from extension
  m_lCulFieldInd = Util.GetFieldIndexes(m_pCulvertLayer.FeatureClass,
"culvert")

  'Setup rasters using the sediment modeler
  success = m_pSedModel.RunRasterAnalysis(m_pRoadLayer.FeatureClass,
m_pDEMLayer.Raster)
  If Not success Then Exit Sub

  'get the newly created distace to streams from the sed model and set
it.
  m_pExtension.DistToStreamsLayer = m_pSedModel.DistanceToStream
  m_pExtension.MaxDeliveryDistance = m_pSedModel.MaxDeliveryDistance
  MsgBox "Raster analysis complete"

  'set the default values given by the user
  m_pSedModel.DefaultRoadAge = m_pExtension.DefaultRoadAge
  m_pSedModel.DefaultRoadGrade = m_pExtension.DefaultRoadGrade
  m_pSedModel.DefaultRoadSurface = m_pExtension.DefaultRoadSurface
  m_pSedModel.DefaultRoadTraffic = m_pExtension.DefaultRoadTraffic
  m_pSedModel.DefaultRoadWidth = m_pExtension.DefaultRoadWidth
  m_pSedModel.DefaultSlopeCover = m_pExtension.DefaultSlopeCover

  If Not m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StartEditing False
  End If

  success = SetSedProd()
  MsgBox "set Sediment production " & success

  success = SetCulSed()
  MsgBox "set Culvert Sediment " & success

  m_pRefresh.Invalidate esriAllScreenCaches
  m_pWksEdit.StopEditing True

  'display total sediment delivered by culverts
  m_pExtension.ShowTotalSed
(Util.SumValuesOnField(m_pCulvertLayer.FeatureClass, _
                                m_lCulFieldInd(3)))

  'release buttons
  m_pExtension.IsAnalyzed = True

  'refresh screen
  m_pDoc.ActiveView.Refresh
  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
```

```
End Sub

Private Function SetCulSed() As Boolean
  On Error GoTo ErrorHandler

  Dim c_lCulIdx As Long
  Dim c_lSedProdx As Long
  Dim c_lSedx As Long
  Dim c_lDelPotx As Long
  c_lCulIdx = m_lCulFieldInd(0)
  c_lSedProdx = m_lRoadFieldInd(10)
  c_lSedx = m_lCulFieldInd(3)
  c_lDelPotx = m_lCulFieldInd(2)

  'get cursor into culvert layer
  Dim pFilter As IQueryFilter
  Set pFilter = New QueryFilter
  Dim pCulCursor As IFeatureCursor
  Set pCulCursor = m_pCulvertLayer.Search(Nothing, False)
  Dim pCul As IFeature
  Dim pPoint As IPoint
  Set pCul = pCulCursor.NextFeature
  Do While Not pCul Is Nothing
    Set pPoint = pCul.Shape
    'set delivery potential if not already set by other operation
    If pCul.Value(c_lDelPotx) <> 1 Then
      pCul.Value(c_lDelPotx) = m_pSedModel.GetDeliveryPotential(pPoint)
    End If
    'set sediment per culvert
    ' !! "CULV" is HARD CODED name and NOT SAFE !!
    pCul.Value(c_lSedx) =
ConnectFnct.SumUpSed(m_pRoadLayer.FeatureClass, c_lSedProdx, _
                     "CULV", pCul.Value(c_lCulIdx)) *
pCul.Value(c_lDelPotx)
    pCul.Store
    Set pCul = pCulCursor.NextFeature
  Loop
  SetCulSed = True

  Exit Function
ErrorHandler:
  SetCulSed = False
  HandleError True, "SetCulSed " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Function SetSedProd() As Boolean
  On Error GoTo ErrorHandler

  Dim c_lSedProdx As Long
  c_lSedProdx = m_lRoadFieldInd(10)

  'get cursor into roads
  Dim pCursor As IFeatureCursor
  Set pCursor = m_pRoadLayer.Search(Nothing, False)
  Dim pRoadSeg As IFeature
  Set pRoadSeg = pCursor.NextFeature
  Do While Not pRoadSeg Is Nothing
```

```
    pRoadSeg.Value(c_lSedProdx) =
m_pSedModel.GetSedimentProduction(pRoadSeg)
    pRoadSeg.Store
    Set pRoadSeg = pCursor.NextFeature
  Loop
  SetSedProd = True

  Exit Function
ErrorHandler:
  SetSedProd = False
  HandleError True, "SetSedProd " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function
```

CLASS – clsSedBox (clsSedBox.cls)

```
Option Explicit

Implements ICommand
Implements IToolControl

Private m_pApp As IApplication
Private WithEvents m_pExt As clsExt

Private Sub Class_Terminate()
  Set m_pApp = Nothing
  Set m_pExt = Nothing
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE

End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "SedBox"
End Property

Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean

End Property

Private Property Get ICommand_Enabled() As Boolean
  'check for certain properties in extension
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted And m_pExt.IsAnalyzed Then
      ICommand_Enabled = True
    Else
      ICommand_Enabled = False
    End If
  Else
    ICommand_Enabled = False
```

```
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long

End Property

Private Property Get ICommand_HelpFile() As String

End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Display Total Sediment"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "TotalSedBox"
End Property

Private Sub ICommand_OnClick()

End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String

End Property

Private Property Get IToolControl_hWnd() As esriCore.OLE_HANDLE
  IToolControl_hWnd = frmTotSed.tboTotSed.hwnd
End Property

Private Function IToolControl_OnDrop(ByVal barType As
esriCore.esriCmdBarType) As Boolean
  If barType = esriCmdBarTypeToolbar Then
      IToolControl_OnDrop = True
  End If
End Function

Private Sub IToolControl_OnFocus(ByVal complete As
esriCore.ICompletionNotify)
  'Set pCompNotify = complete
  complete.SetComplete
End Sub

Private Sub m_pExt_IsStopping()
  frmTotSed.tboTotSed.Text = ""
End Sub

Private Sub m_pExt_ShowSediment(amount As Double)
  frmTotSed.tboTotSed.Text = Math.Round(amount, 3)
End Sub
```

CLASS – clsSegGrade (clsSegGrade.cls)

```
Option Explicit

Implements ICommand
Implements ITool

Private m_pApp As IApplication
Private m_pBitmap As IPictureDisp
Private WithEvents m_pExt As clsExt
Private m_pFeatClass As IFeatureClass
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea
Private m_lGradeIndex As Long
Private m_pNewPoint As IPoint
Private m_pDisplay As IDisplay
Private m_frmGF As frmGradeField
Private m_pFeatSel As IFeatureSelection
Private m_pFeature As IFeature

Private Sub Class_Initialize()
  'load the button image from the resource file
  Set m_pBitmap = LoadResPicture("Grade", vbResBitmap)
  m_lGradeIndex = -1
End Sub

Private Sub Class_Terminate()
  Set m_pBitmap = Nothing
  Set m_pExt = Nothing
  Set m_pApp = Nothing
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  ICommand_Bitmap = m_pBitmap
End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "SegmentGrade"
End Property

Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean
  'TODO: your implementation here
End Property

Private Property Get ICommand_Enabled() As Boolean
  'check for certain properties in extension
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted And m_pExt.IsSetUp And Not m_pExt.HasTopology
Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
```

```
    End If
  Else: ICommand_Enabled = False
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long
  'TODO: your implementation here
End Property

Private Property Get ICommand_HelpFile() As String
  'TODO: your implementation here
End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Adjust Road Grade"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "Grade"
End Property

Private Sub ICommand_OnClick()
  On Error GoTo erh
  Dim pMxDoc As IMxDocument
  Set pMxDoc = m_pApp.Document
  Set m_pDisplay = pMxDoc.ActiveView.ScreenDisplay
  Set m_pFeatClass = m_pExt.RoadLayer.FeatureClass
  'set selcetion
  Set m_pFeatSel = m_pExt.RoadLayer
  'get the workspace to edit
  Dim pDS As IDataset
  Set pDS = m_pFeatClass
  Set m_pWksEdit = pDS.Workspace
  If Not m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StartEditing True
  End If
  'get grade field index
  m_lGradeIndex = m_pFeatClass.FindField(m_pExt.GradeName)

  'show form
  Set m_frmGF = New frmGradeField
  'move to right hand corner
  m_frmGF.Show vbModeless

  'create new screen refresh
  Set m_pRefresh = New InvalidArea
  Set m_pRefresh.Display = m_pDisplay

  Exit Sub
erh:
  MsgBox "error in create SegGrade cmd: " & Error
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)
End Sub
```

```
Private Property Get ICommand_Tooltip() As String
  ICommand_Tooltip = "Manualy Adjust A Road Segment's Grade "
End Property

Private Property Get ITool_Cursor() As esriCore.OLE_HANDLE

End Property

Private Function ITool_Deactivate() As Boolean
  Set m_pNewPoint = Nothing
  Set m_pDisplay = Nothing
  Set m_pFeature = Nothing
  Set m_pFeatSel = Nothing
  Set m_pFeatClass = Nothing

  If Not m_pWksEdit Is Nothing Then
    m_pWksEdit.StopEditing True
  End If
  Set m_pWksEdit = Nothing

  m_frmGF.Hide
  'destroy the form
  Unload m_frmGF
  Set m_frmGF = Nothing

  ITool_Deactivate = True
End Function

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long)
As Boolean
  ITool_OnContextMenu = True
End Function

Private Sub ITool_OnDblClick()

End Sub

Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
 On Error GoTo erh
  If Button = 1 Then
    'make point
    Set m_pNewPoint = m_pDisplay.DisplayTransformation.ToMapPoint(X, Y)
    'get feature that intesects point
    Set m_pFeature = Util.FindFeatureNearPoint(m_pFeatClass,
m_pNewPoint, 10)
    If m_pFeature Is Nothing Then Exit Sub
    'get grade value of this feature
    Dim iGradeVal As Integer
    iGradeVal = m_pFeature.Value(m_lGradeIndex)
```

```
    'select this feature
    m_pFeatSel.Clear
    m_pFeatSel.Add m_pFeature
    m_pRefresh.Add m_pFeature
    m_pRefresh.Invalidate esriAllScreenCaches
    'set up form and set it's position
    m_frmGF.GradeValue = iGradeVal
    m_frmGF.SetFocus
    Exit Sub
  Else 'button is 2
    If Not m_pFeature Is Nothing Then
      'get it's value
      If IsNumeric(m_frmGF.GradeValue) Then
        'set the feature's grade to this value
        m_pWksEdit.StartEditOperation
        m_pFeature.Value(m_lGradeIndex) = CInt(m_frmGF.GradeValue)
        m_pFeature.Store
        m_pWksEdit.StopEditOperation
        'deselect feature
        m_pFeatSel.Clear
        m_pRefresh.Invalidate esriAllScreenCaches
        m_frmGF.GradeValue = ""
      End If
    End If
  End If
  Exit Sub
erh:
  MsgBox "error " & Error
End Sub

Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  'unused
End Sub

Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  'not used
End Sub

Private Sub ITool_Refresh(ByVal hDC As esriCore.OLE_HANDLE)
  'unused
End Sub

Private Sub m_pExt_IsStopping()
  If Not m_frmGF Is Nothing Then
    m_frmGF.Hide
    Set m_frmGF = Nothing
  End If
  If Not m_pWksEdit Is Nothing Then
    If m_pWksEdit.IsBeingEdited Then
      m_pWksEdit.StopEditing True
    End If
  End If
End Sub
```

CLASS - clsSetUpFlow (clsSetUpFlow.cls)

```
Option Explicit

Implements ICommand

Private m_pApp As IApplication
Private m_pDoc As IMxDocument
Private m_pExtension As clsExt
Private m_pRoadLayer As IFeatureLayer
Private m_pStreamLayer As IFeatureLayer
Private m_pCulvertLayer As IFeatureLayer

Private WithEvents m_pFrm As frmChildDec
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea
Private m_colUserDec As Collection
Private m_lRoadFieldInd() As Long
Private m_lCulFieldInd() As Long

Private m_ErLog() As Long

' Variables used by the Error handler function - DO NOT REMOVE
Const c_ModuleFileName =
"C:\Evenflo\ThesisWorks\VBScripts\CrossDrainSpacer\clsSetUpFlow.cls"

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  On Error GoTo ErrorHandler

  'TODO: your implementation here

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

  ICommand_Caption = "Flow SetUp"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

  ICommand_Category = "Road Sediment Analyst"

  Exit Property
ErrorHandler:
```

```
  HandleError True, "ICommand_Category " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  'TODO: your implementation here

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

  If Not m_pExtension Is Nothing Then
    If m_pExtension.IsSetUp And (Not m_pExtension.HasTopology) Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  'TODO: your implementation here

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  'TODO: your implementation here

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property
```

```
Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

  ICommand_Message = "Setup the Ditch Water Flow of a Road System"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Message " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

  ICommand_Name = "Flow_SetUp"

  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler

  'disable buttons
  m_pExtension.HasTopology = False

  'get layers from extension
  Set m_pRoadLayer = m_pExtension.RoadLayer
  Set m_pStreamLayer = m_pExtension.StreamLayer
  Set m_pCulvertLayer = m_pExtension.CulvLayer

  Dim success As Boolean
  Dim pRoadCulvSelSet As ISelectionSet

  Dim pWks As IWorkspace
  Dim pDS As IDataset
  Set pDS = m_pRoadLayer
  Set pWks = pDS.Workspace
  Set m_pWksEdit = pWks

  '1
  If m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StopEditing True
  End If
  success = Util.CreateRSAFields(m_pRoadLayer.FeatureClass, _
                                 m_pCulvertLayer.FeatureClass)
  'get field indexes in arrays
  m_lRoadFieldInd = Util.GetFieldIndexes(m_pRoadLayer.FeatureClass,
"road")
  m_lCulFieldInd = Util.GetFieldIndexes(m_pCulvertLayer.FeatureClass,
"culvert")
  MsgBox "created fields " & success

  '2
  If Not m_pWksEdit.IsBeingEdited Then
```

```
    m_pWksEdit.StartEditing False
  End If
  'give unique identifiers to road segments
  success = SetUniqueID()
  MsgBox "set unique ids " & success

  '2
  m_pWksEdit.StartEditing False
  success = SetStreamCross()
  'm_pWKSEdit.StopEditing True
  m_pRefresh.Invalidate esriAllScreenCaches
  MsgBox "set up stream intersections" & success

  '3
  m_ErLog = SplitRoadsAtExistingCulverts()
  MsgBox "set up existing culverts"

  '4
  'm_pWKSEdit.StartEditing False
  success = SetEndPointsId()
  MsgBox "set end points id " & success
  'm_pWKSEdit.StopEditing True

  '5
  success = CheckDownStreamConnect()

  If success Then
    success = SetNetworkTopology()
    DisplayErrorReport success
  End If


  Exit Sub
ErrorHandler:
  'this is needed to stop editing wks if anything went wrong
  m_pWksEdit.StopEditing False
  HandleError True, "ICommand_OnClick " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

  Set m_pApp = hook
  Set m_pDoc = m_pApp.Document
  Set m_pRefresh = New InvalidArea
  Set m_pRefresh.Display = m_pDoc.ActiveView.ScreenDisplay
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExtension = m_pApp.FindExtensionByCLSID(pId)


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub
```

```
Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler

  ICommand_Tooltip = "Run The Analysis For Placing Culvert Operations"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

'has to run inside an edit session.
Private Function SetUniqueID() As Boolean
  Dim pFeatCur As IFeatureCursor
  Dim pFeat As IFeature
  Dim lCounter As Long
  Dim lFIndex As Long

  On Error GoTo ErrorHandler

  'Select all roads in the layer
  Set pFeatCur = m_pRoadLayer.Search(Nothing, False)
  Set pFeat = pFeatCur.NextFeature
  lCounter = 1
  lFIndex = m_lRoadFieldInd(0)

  If lFIndex = -1 Then
    MsgBox "Required Field not found !", vbCritical
    Exit Function
  End If

  Do While Not pFeat Is Nothing
    pFeat.Value(lFIndex) = lCounter
    pFeat.Store
    lCounter = lCounter + 1

    Set pFeat = pFeatCur.NextFeature
  Loop

  SetUniqueID = True
  Exit Function

ErrorHandler:
  SetUniqueID = False
  HandleError True, "SetUniqueID " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4

End Function

'has to run inside edit session
Private Function SetEndPointsId() As Boolean
  On Error GoTo ErrorHandler

  'select all features in m_pRoadLayer
  Dim pFeatCur As IFeatureCursor
```

```
  Set pFeatCur = m_pRoadLayer.Search(Nothing, False)

  Dim pRoadSeg As IFeature
  Set pRoadSeg = pFeatCur.NextFeature
  Dim pPolyline As IPolyline
  Dim pEndPoints As IPointCollection
  Set pEndPoints = New Multipoint

   'get required field indexes
  Dim lFPIdx As Long, lTPIdx As Long
  lFPIdx = m_lRoadFieldInd(3)
  lTPIdx = m_lRoadFieldInd(4)

  'get max point number
  Dim lMaxPointId As Long
  lMaxPointId = Util.GetMaxOfFields(m_pRoadLayer.FeatureClass, lFPIdx,
lTPIdx)

  'add all end points to a collection
  Do While Not pRoadSeg Is Nothing
    Set pPolyline = pRoadSeg.Shape
    pEndPoints.AddPoint pPolyline.FromPoint
    pEndPoints.AddPoint pPolyline.ToPoint
    Set pRoadSeg = pFeatCur.NextFeature
  Loop

  'symplify if necessary
  Dim pTopoOp As ITopologicalOperator
  Set pTopoOp = pEndPoints
  If Not pTopoOp.IsSimple Then pTopoOp.Simplify

  Dim pPoint As IPoint
  Dim pFilter As ISpatialFilter
  Set pFilter = New SpatialFilter
  pFilter.SpatialRel = esriSpatialRelTouches
  'go through each point and set road from and to ids
  Dim counter As Integer
  For counter = 0 To pEndPoints.PointCount - 1
    Set pPoint = pEndPoints.Point(counter)
    Set pFilter.Geometry = pPoint
    Set pFeatCur = m_pRoadLayer.Search(pFilter, False)
    Set pRoadSeg = pFeatCur.NextFeature
    Do While Not pRoadSeg Is Nothing
      Set pPolyline = pRoadSeg.Shape
      If pPolyline.FromPoint.Compare(pPoint) = 0 Then
        If pRoadSeg.Value(lFPIdx) = 0 Then
          pRoadSeg.Value(lFPIdx) = lMaxPointId + counter + 1
          pRoadSeg.Store
        End If
      Else
        If pRoadSeg.Value(lTPIdx) = 0 Then
          pRoadSeg.Value(lTPIdx) = lMaxPointId + counter + 1
          pRoadSeg.Store
        End If
      End If
      Set pRoadSeg = pFeatCur.NextFeature
    Loop
  Next counter
```

```
  SetEndPointsId = True
  Exit Function

ErrorHandler:
  SetEndPointsId = False
  HandleError True, "SetEndPointsID " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4

End Function

Private Function SetAllParents() As Boolean
  On Error GoTo ErrorHandler

  Dim lFromNode As Long
  Dim pRoad As IFeature
  Dim lParents() As Long
  Dim lChildren() As Long
  Dim lRoadFrom As Long
  Dim lRoadTo As Long
  Dim lTemp As Long
  ReDim undecidedChild(0) As Long

  'get required field indexes
  Dim lFPIdx As Long, lIDIdx As Long, lTPIdx As Long
  Dim lP1Idx As Long, lP2Idx As Long, lP3Idx As Long
  Dim lChIdx As Long
  lFPIdx = m_lRoadFieldInd(3)
  lIDIdx = m_lRoadFieldInd(0)
  lTPIdx = m_lRoadFieldInd(4)
  lP1Idx = m_lRoadFieldInd(5)
  lP2Idx = m_lRoadFieldInd(6)
  lP3Idx = m_lRoadFieldInd(7)
  lChIdx = m_lRoadFieldInd(8)

  'select all roads
  Dim pFeatCur As IFeatureCursor
  Set pFeatCur = m_pRoadLayer.Search(Nothing, False)

  Set pRoad = pFeatCur.NextFeature
  Do While Not pRoad Is Nothing
    lRoadFrom = pRoad.Value(lFPIdx)
    lRoadTo = pRoad.Value(lTPIdx)
    'get parents
    lParents() = FindConnected(lIDIdx, "TOPT = " & lRoadFrom)
    'set parents only if they were not forced -1 before (inserting a
culvert could do that)
    If pRoad.Value(lP1Idx) <> -1 Then pRoad.Value(lP1Idx) = lParents(0)
    If pRoad.Value(lP2Idx) <> -1 Then pRoad.Value(lP2Idx) = lParents(1)
    If pRoad.Value(lP3Idx) <> -1 Then pRoad.Value(lP3Idx) = lParents(2)
    'get children
    lChildren() = FindConnected(lIDIdx, "FROMPT = " & lRoadTo)
    lTemp = ConnectFnct.CheckUniqueChild(lChildren)
    'set child only if a child was not forced -1 before (when inserting
a culvert)
    If lTemp >= -1 Then
      If pRoad.Value(lChIdx) <> -1 Then pRoad.Value(lChIdx) = lTemp
    Else
      'this should be taken out
```

```
      'check downstream connect has already fixed the multiple children
problem
      MsgBox "Network Topology Error: found more than one child at
segment " & pRoad.Value(lIDIdx)
    End If

    pRoad.Store
    Set pRoad = pFeatCur.NextFeature
  Loop

  SetAllParents = True
  Exit Function
ErrorHandler:
  SetAllParents = False
  HandleError True, "SetAllParents " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Function FindConnected(lIDIdx As Long, sQueryString As String)
As Long()
  On Error GoTo ErrorHandler

  Dim connectedID(2) As Long
  Dim pConSeg As IFeature
  Dim pFeatCur As IFeatureCursor

  'define query
  Dim pQuery As IQueryFilter
  Set pQuery = New QueryFilter
  pQuery.WhereClause = sQueryString

  'search for roads that end with the input road from point
  Set pFeatCur = m_pRoadLayer.Search(pQuery, False)
  Set pConSeg = pFeatCur.NextFeature 'pConSeg is now the first connected
segment

  'loop three times. Three connections maximum
  Dim i As Integer
  For i = 0 To 2
    If Not pConSeg Is Nothing Then
      connectedID(i) = pConSeg.Value(lIDIdx)
    Else: connectedID(i) = -1
    End If
    Set pConSeg = pFeatCur.NextFeature
  Next i

  FindConnected = connectedID


  Exit Function
ErrorHandler:
  HandleError False, "FindConnected " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Function SetCulverts() As Boolean
  On Error GoTo ErrorHandler
```

```
  Dim lTPIdx As Long, lCuIdx As Long
  lTPIdx = m_lRoadFieldInd(4)
  lCuIdx = m_lRoadFieldInd(9)

  Dim maxCulvID As Long
  Dim curCulvID As Long
  Dim lCRSAID As Long
  lCRSAID = m_lCulFieldInd(0)
  maxCulvID = Util.GetMaxValue(m_pCulvertLayer.FeatureClass, lCRSAID)
  If maxCulvID = -2147483648# Then
    maxCulvID = 0
  End If
  Debug.Print "max culv no is now : " & maxCulvID

  Dim lP1x As Long, lP2x As Long, lP3x As Long
  lP1x = m_lRoadFieldInd(5)
  lP2x = m_lRoadFieldInd(6)
  lP3x = m_lRoadFieldInd(7)

  'select all features with no child
  Dim pCulvSelSet As ISelectionSet
  Dim pFilter As IQueryFilter
  Set pFilter = New QueryFilter
  pFilter.WhereClause = "CHILD = -1"
  Dim pRoadDS As IDataset
  Set pRoadDS = m_pRoadLayer
  Set pCulvSelSet = m_pRoadLayer.FeatureClass.Select(pFilter, _
                    esriSelectionTypeHybrid, esriSelectionOptionNormal,
_
                    pRoadDS.Workspace)
  Dim pFeatCur As IFeatureCursor
  pCulvSelSet.Search Nothing, False, pFeatCur
  Dim pFeat As IFeature
  Set pFeat = pFeatCur.NextFeature

  Debug.Print "child = -1 count = " & pCulvSelSet.count

  'Loop and select all features that flow to same point
  Dim pToFilter As IQueryFilter
  Set pToFilter = New QueryFilter
  Dim pToFeatCur As IFeatureCursor
  Dim pSameToFeat As IFeature
  Dim pNewCulv As IFeature
  Dim pPolyline As IPolyline
  Dim pExistingCulvert As IFeature
  Dim existingRSAID As Long

  Do While Not pFeat Is Nothing
    Debug.Print pFeat.OID
    pToFilter.WhereClause = "TOPT = " & pFeat.Value(lTPIdx)
    pCulvSelSet.Search pToFilter, False, pToFeatCur

    Set pSameToFeat = pToFeatCur.NextFeature
    'add a culvert to the culvert layer with its RSAID specified here
    If Not pSameToFeat Is Nothing Then
      'see if it there already is a culvert at the TOPT location
      Set pPolyline = pSameToFeat.Shape
```

```
      Set pExistingCulvert =
Util.FindFeatureNearPoint(m_pCulvertLayer.FeatureClass,
                          pPolyline.ToPoint, 0.000000001)

      If Not pExistingCulvert Is Nothing Then
        'there is no need to create a new culvert feature; use this one
        'check to see if existing culvert already has an RSAID value
        existingRSAID = pExistingCulvert.Value(lCRSAID)
        If existingRSAID > 0 Then
          'there is no need for a new ID entry; use this one
          curCulvID = existingRSAID 'this value is used below!!!!
        Else
          'must create new id for existing culvert
          maxCulvID = maxCulvID + 1
          curCulvID = maxCulvID 'this value is used below!!!!!
          pExistingCulvert.Value(lCRSAID) = curCulvID
          pExistingCulvert.Store
        End If
      Else 'Could not find a feature in the culvert layer at the
specific location
        'must create a new culvert
        Set pNewCulv = m_pCulvertLayer.FeatureClass.CreateFeature
        Set pNewCulv.Shape = pPolyline.ToPoint
        'must create new id for the new culvert
        maxCulvID = maxCulvID + 1
        curCulvID = maxCulvID 'this value is used below!!!!!
        pNewCulv.Value(lCRSAID) = curCulvID
        pNewCulv.Store
        m_pRefresh.Add pNewCulv
      End If
    End If

    'go through the all features that flow to same pt and set them to
flow to this culvert
    Do While Not pSameToFeat Is Nothing
      pSameToFeat.Value(lCuIdx) = curCulvID
      pSameToFeat.Store
      'recursevly follow upstream and set all parents to this culvert
      ConnectFnct.SetUpstream curCulvID, m_pRoadLayer, pSameToFeat,
lP1x, lP2x, lP3x, lCuIdx
      pCulvSelSet.RemoveList 1, pSameToFeat.OID
      Debug.Print "selection has " & pCulvSelSet.count
      Set pSameToFeat = pToFeatCur.NextFeature
    Loop

    Set pFeat = pFeatCur.NextFeature
  Loop

  SetCulverts = True
  Exit Function
ErrorHandler:
  SetCulverts = False
  HandleError True, "SetCulverts " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Public Function SetStreamCross() As Boolean
  On Error GoTo ErrorHandler
```

```
  Dim lP1x As Long, lP2x As Long, lP3x As Long, lChx As Long, lIdx As
Long, lFpx As Long, lTpx As Long
  lP1x = m_lRoadFieldInd(5)
  lP2x = m_lRoadFieldInd(6)
  lP3x = m_lRoadFieldInd(7)
  lChx = m_lRoadFieldInd(8)
  lIdx = m_lRoadFieldInd(0)
  lTpx = m_lRoadFieldInd(4)
  lFpx = m_lRoadFieldInd(3)

  'get the maximum existing point number in the road table
  Dim lMaxPointId As Long
  lMaxPointId = Util.GetMaxOfFields(m_pRoadLayer.FeatureClass, lFpx,
lTpx)

  Dim lMaxId As Long
  lMaxId = Util.GetMaxValue(m_pRoadLayer.FeatureClass, lIdx)
  'when there are no features exit
  If lMaxId = -2147483648# Then
    MsgBox "no roads are present"
    Exit Function
  End If
  Debug.Print "max road RSAID is" & lMaxId

  'get name of the shape field
  Dim sShapeName As String
  sShapeName = m_pRoadLayer.FeatureClass.ShapeFieldName
  'Select all Roads
  Dim pRoadCur As IFeatureCursor
  Set pRoadCur = m_pRoadLayer.Search(Nothing, False)
  'Spatial filter for intersections
  Dim pSpFilter As ISpatialFilter
  Set pSpFilter = New SpatialFilter
  Dim pStreamCur As IFeatureCursor
  Dim pStream As IFeature
  With pSpFilter
    .GeometryField = sShapeName
    .SpatialRel = esriSpatialRelIntersects
  End With
  'instantiate a stream feature selection
  Dim pStreamFeatSel As IFeatureSelection
  Set pStreamFeatSel = m_pStreamLayer
  pStreamFeatSel.Clear
  'get first road
  Dim pRoad As IFeature
  Set pRoad = pRoadCur.NextFeature
  Do While Not pRoad Is Nothing
    'set spatial filter to current road
    Set pSpFilter.Geometry = pRoad.Shape
    'search for Streams that intersect curent road
    Set pStreamCur = m_pStreamLayer.Search(pSpFilter, True)
    Set pStream = pStreamCur.NextFeature
    Do While Not pStream Is Nothing
      pStreamFeatSel.Add pStream
      'split road at intersection point
      Set pRoad.Shape = SplitRoadAtPoint(pStream.Shape, pRoad.Shape)
      Set pStream = pStreamCur.NextFeature
    Loop
```

```
    'build new features for all segments in the split
    SplitRoadWithStream pStreamFeatSel, pRoad, lMaxId, lMaxPointId,
lIdx, lTpx, lFpx
    'delete original segment
    pRoad.Delete
    pStreamFeatSel.Clear
    Set pRoad = pRoadCur.NextFeature
  Loop

  SetStreamCross = True
  Exit Function
ErrorHandler:
  SetStreamCross = False
  HandleError True, "SetStreamCross " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4

End Function
'make a new polyline made of segments resulting from intersecting with
stream
Private Function SplitRoadAtPoint(ByRef pStreamPoly As IPolyline, _
                                  ByRef pRoadPoly As IPolyline) As IPolyline
  On Error GoTo ErrorHandler


  Dim pTopoOp As ITopologicalOperator
  Dim pLeftPiece As IPolyline
  Dim pRightPiece As IPolyline
  Dim pNewPoly As IPolyline
  Set pNewPoly = New Polyline
  Dim pNPColl As IGeometryCollection
  Set pNPColl = pNewPoly
  Dim pPartColl As IGeometryCollection

  Dim pPolyColl As IGeometryCollection
  Set pPolyColl = New GeometryBag
  Util.BreakPolyIntoPolySegments pRoadPoly, pPolyColl
  Debug.Print "input segments : " & pPolyColl.GeometryCount
  Dim b As Boolean
  Dim i As Integer
  For i = 0 To pPolyColl.GeometryCount - 1
    Set pTopoOp = pPolyColl.Geometry(i)
    b = pTopoOp.IsSimple
    pTopoOp.Cut pStreamPoly, pLeftPiece, pRightPiece
    If Not pLeftPiece Is Nothing And Not pRightPiece Is Nothing Then
      If pLeftPiece.Length > 0 And pRightPiece.Length > 0 Then
        Set pPartColl = pLeftPiece
        Debug.Print "left part segments : " & pPartColl.GeometryCount
        pNPColl.AddGeometryCollection pPartColl
        Set pPartColl = pRightPiece
        Debug.Print "right part segments : " & pPartColl.GeometryCount
        pNPColl.AddGeometryCollection pPartColl
      Else
        Set pPartColl = pPolyColl.Geometry(i)
        pNPColl.AddGeometryCollection pPartColl
      End If
    Else
      Set pPartColl = pPolyColl.Geometry(i)
      pNPColl.AddGeometryCollection pPartColl
```

```
    End If
  Next i
  Set SplitRoadAtPoint = pNewPoly
  Debug.Print "geomcount is :" & pNPColl.GeometryCount

  Set pTopoOp = Nothing
  Set pLeftPiece = Nothing
  Set pRightPiece = Nothing
  Set pNewPoly = Nothing
  Set pNPColl = Nothing
  Set pNPColl = Nothing
  Set pPartColl = Nothing
  Set pPolyColl = Nothing

  Exit Function
ErrorHandler:
  HandleError False, "SplitRoadAtPoint " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Sub SplitRoadWithStream(ByRef pStreamSel As IFeatureSelection,
ByRef pRoad As IFeature, _
                                ByRef lMaxId As Long, ByRef lMaxPointId
As Long, _
                                lIdx As Long, lTpx As Long, lFpx As
Long)
  On Error GoTo ErrorHandler

  'prepare selection to hold polylines
  Dim pRoadSel As IFeatureSelection
  Set pRoadSel = m_pRoadLayer
  pRoadSel.Clear
  'get segments in Road and make them polylines
  Dim pGeomColl As IGeometryCollection
  Set pGeomColl = New GeometryBag
  Util.BreakPolyIntoPolySegments pRoad.Shape, pGeomColl

  Dim pFeatClass As IFeatureClass
  Set pFeatClass = m_pRoadLayer.FeatureClass

  Dim i As Long
  For i = 0 To pGeomColl.GeometryCount - 1
    Dim pFeat As IFeature
    Set pFeat = pFeatClass.CreateFeature
   ' MsgBox "created feat " & pFeat.OID
    Util.CopyAllAtributes pRoad, pFeat
   ' MsgBox "copied attrib from " & pRoad.OID & " to " & pFeat.OID
    Set pFeat.Shape = pGeomColl.Geometry(i)
    lMaxId = lMaxId + 1
    pFeat.Value(lIdx) = lMaxId
    pFeat.Store
   ' MsgBox "updated shape and id for " & pFeat.OID
    'add feat to collection
    pRoadSel.Add pFeat
    m_pRefresh.Add pFeat
   ' MsgBox "refreshed"
  Next i
```

```
  'insert an artificial break in the road table either at frompt or topt
  MakeFakeBreak pRoadSel, pStreamSel, lMaxPointId, lFpx, lTpx, True
 ' MsgBox "made fake break"
  'relese memory
  Set pGeomColl = Nothing

  Exit Sub
ErrorHandler:
  HandleError True, "SplitRoadWithStream " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub
Private Function MakeFakeBreak(ByRef pRoadSel As IFeatureSelection,
ByRef pStreamSel As IFeatureSelection, _
                          ByRef lMaxPointId As Long, lFpx As Long, lTpx
As Long, flag As Boolean) As Boolean
  On Error GoTo ErrorHandler


  Dim success As Boolean
  success = False
  Dim pStreamSet As ISelectionSet
  Set pStreamSet = pStreamSel.SelectionSet
  Dim pStreamCur As IFeatureCursor
  'select all streams in given set
  pStreamSet.Search Nothing, False, pStreamCur
  Dim pStream As IFeature
  Set pStream = pStreamCur.NextFeature
  Dim pSpFilter As ISpatialFilter
  Set pSpFilter = New SpatialFilter
  pSpFilter.SpatialRel = esriSpatialRelTouches

  Dim pRoadSet As ISelectionSet
  Set pRoadSet = pRoadSel.SelectionSet
  Dim pRoadCur As IFeatureCursor
  Dim pRoad As IFeature

  Dim pRoadPoly As IPolyline
  Dim pEndPoint As IPoint
  Dim pStreamPoly As IPolyline
  Dim pRelOp As IRelationalOperator

  Do While Not pStream Is Nothing
    Set pStreamPoly = pStream.Shape
    'get the road segments from the roadset touching pStream
    Set pSpFilter.Geometry = pStreamPoly
    pRoadSet.Search pSpFilter, False, pRoadCur

    Set pRoad = pRoadCur.NextFeature
    Do While Not pRoad Is Nothing
      'identify end that touches
      Set pRoadPoly = pRoad.Shape
      'try from end first
      Set pEndPoint = pRoadPoly.FromPoint
      Set pRelOp = pEndPoint
      If pRelOp.Within(pStreamPoly) Then
        pRoad.Value(lFpx) = lMaxPointId + 1 ' set flow "FROM" to this
point
        pRoad.Store
```

```
        lMaxPointId = lMaxPointId + 1
      End If
      'try the "to" end
      Set pEndPoint = pRoadPoly.ToPoint
      Set pRelOp = pEndPoint
      If pRelOp.Within(pStreamPoly) Then
        pRoad.Value(lTpx) = lMaxPointId + 1 ' set flow "TO" to this
point
        pRoad.Store
        lMaxPointId = lMaxPointId + 1
      End If
      'iterate
      Set pRoad = pRoadCur.NextFeature
    Loop
  'iterate streams
  Set pStream = pStreamCur.NextFeature
  Loop

  'release memory
  Set pStreamSet = Nothing
  Set pStream = Nothing
  Set pRoadSet = Nothing
  Set pSpFilter = Nothing
  Set pRoad = Nothing
  Set pStreamCur = Nothing
  Set pRoadCur = Nothing
  Set pRelOp = Nothing
  Set pEndPoint = Nothing

  success = True
  'returns true if all break have been written
  MakeFakeBreak = success


  Exit Function
ErrorHandler:
  HandleError True, "MakeFakeBreak " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function

Private Sub m_pFrm_HasFinished(bCancel As Boolean)
  On Error GoTo ErrorHandler

  'for all user choices go through their unwanted children and cut them
off
  'artificial break point at their FROMPT
  If Not bCancel Then
    'find indexes
    Dim lIdx As Long, lFpx As Long, lTpx As Long
    lIdx = m_lRoadFieldInd(0)
    lFpx = m_lRoadFieldInd(3)
    lTpx = m_lRoadFieldInd(4)
    'find max value from fields
    Dim lMaxPointValue As Long
    lMaxPointValue = Util.GetMaxOfFields(m_pRoadLayer.FeatureClass,
lFpx, lTpx)

    'm_pWKSEdit.StartEditing False
```

```
    Dim pFeatCur As IFeatureCursor
    Dim pFeat As IFeature
    Dim pFilter As IQueryFilter
    Set pFilter = New QueryFilter

    Dim counter As Integer
    Dim colChildren As Collection
    For Each colChildren In m_colUserDec
      MsgBox colChildren.count - 1   'verification ###### take out
      For counter = 2 To colChildren.count
        'find feature
        pFilter.WhereClause = "RSAID = " & colChildren.Item(counter)
        Set pFeatCur = m_pRoadLayer.Search(pFilter, False)
        Set pFeat = pFeatCur.NextFeature
        If Not pFeat Is Nothing Then
          'modify "from point" value
          lMaxPointValue = lMaxPointValue + 1
          pFeat.Value(lFpx) = lMaxPointValue
          pFeat.Store
        End If
      Next counter
    Next

    'm_pWKSEdit.StopEditing True

    'continue set up
    Dim success As Boolean
    success = SetNetworkTopology

    'display message
    DisplayErrorReport success
  Else
    'user has canceled the dialog
    'do not save edits
    m_pWksEdit.StopEditing False
    m_pDoc.ActiveView.PartialRefresh esriViewGeography +
esriViewGraphics, Nothing, Nothing
  End If

  'release memory
  Set pFilter = Nothing
  Set colChildren = Nothing
  Exit Sub
ErrorHandler:
  'this is needed to stop editing wks if anything went wrong
  m_pWksEdit.StopEditing False
  HandleError True, "m_pr_HasFinished " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4

End Sub

'go through each segment in the road layer and see if any segment has
'it's end point as start point for more than one other segment
'if yes add the segment to a collection and have user decide

Private Function CheckDownStreamConnect() As Boolean
  On Error GoTo ErrorHandler
```

```
  'find indexes
  Dim lIdx As Long, lFpx As Long, lTpx As Long
  lIdx = m_lRoadFieldInd(0)
  lFpx = m_lRoadFieldInd(3)
  lTpx = m_lRoadFieldInd(4)
  Dim colChildren As Collection
  Set m_colUserDec = New Collection

  'select all roads
  Dim pWks As IWorkspace
  Set pWks = m_pWksEdit
  Dim lCurTP As Long
  Dim pFilter As IQueryFilter
  Set pFilter = New QueryFilter
  Dim pSelection As ISelectionSet
  Dim pFeatCur As IFeatureCursor
  Set pFeatCur = m_pRoadLayer.Search(Nothing, False)
  Dim pChildrenCur As IFeatureCursor
  Dim pChild As IFeature

  'loop through all
  Dim pFeat As IFeature
  Set pFeat = pFeatCur.NextFeature
  Do While Not pFeat Is Nothing
    'get end pt
    lCurTP = pFeat.Value(lTpx)
    'select all features that have from point equal to lCurEndPt
    pFilter.WhereClause = "FROMPT = " & lCurTP
    Set pSelection = m_pRoadLayer.FeatureClass.Select(pFilter,
esriSelectionTypeIDSet, _

esriSelectionOptionNormal, pWks)
    If pSelection.count > 1 Then
      'add the features rsaid as first element in children's collection
      Set colChildren = New Collection
      colChildren.Add pFeat.Value(lIdx)
      'add the children starting with element 2 of the children
collection
      pSelection.Search Nothing, False, pChildrenCur
      Set pChild = pChildrenCur.NextFeature
      Do While Not pChild Is Nothing
        colChildren.Add pChild.Value(lIdx)
        'move to next
        Set pChild = pChildrenCur.NextFeature
      Loop
      'add the children collection to user decision collection
      m_colUserDec.Add colChildren
      Set colChildren = Nothing
    End If

    'move to next
    Set pFeat = pFeatCur.NextFeature
  Loop

  If m_colUserDec.count > 0 Then
    Set m_pFrm = New frmChildDec
    Dim pMap As IMap
    Set pMap = m_pDoc.FocusMap
```

```
    m_pFrm.SetUpDialog m_colUserDec, m_pRoadLayer, pMap
    m_pFrm.Show vbModeless
    CheckDownStreamConnect = False
    Exit Function
  End If

  CheckDownStreamConnect = True

  Exit Function
ErrorHandler:
  HandleError False, "CheckDownStreamConnect " & c_ModuleFileName & " "
& GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

Private Function SetNetworkTopology() As Boolean
  On Error GoTo ErrorHandler

  Dim success As Boolean
  '6
  'm_pWKSEdit.StartEditing False
  success = SetAllParents()
  m_pRefresh.Invalidate esriAllScreenCaches
  MsgBox "Set Parents " & success
  'm_pWKSEdit.StopEditing True

  '7
  'm_pWKSEdit.StartEditing False
  success = SetCulverts()
  MsgBox "set all culverts " & success

  '8
  success = EvaluateCulvertCode()
  MsgBox "evaluated culvert removal options"

  m_pRefresh.Invalidate esriAllScreenCaches
  m_pWksEdit.StopEditing True

  'release buttons
  m_pExtension.HasTopology = True

  SetNetworkTopology = True

  Exit Function
ErrorHandler:
  m_pWksEdit.StopEditing False
  SetNetworkTopology = False
  HandleError True, "SetNetworkTopology " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function
Private Function SplitRoadsAtExistingCulverts() As Long()
  On Error GoTo ErrorHandler

  ReDim ErrorLog(0) As Long
  Dim count As Integer
  count = 0
  Dim removeCode As Integer
```

```
  'get remove code field index from culvert layer
  Dim rmcdIDx As Long
  rmcdIDx = m_lCulFieldInd(1)

  'MsgBox "debug 1"

  'create snap agent into the roads layer
  Dim pRoadSnapper As ISnapAgent
  Set pRoadSnapper =
Util.CreateBoundarySnapAgent(m_pRoadLayer.FeatureClass)

  'MsgBox "debug 2"

  'select all culverts from the culvert layer
  Dim pCulvCursor As IFeatureCursor
  Set pCulvCursor = m_pCulvertLayer.Search(Nothing, False)
  Dim pExistingCulvert As IFeature
  Set pExistingCulvert = pCulvCursor.NextFeature
  Do While Not pExistingCulvert Is Nothing
    'MsgBox "examining culvert: " & pExistingCulvert.OID
    'go through all culverts modifying road geometries and topological
attributes
    removeCode = InsertExistingCulvertLocation(pExistingCulvert,
m_pRoadLayer.FeatureClass, _
                                pRoadSnapper, 100, m_lRoadFieldInd)

    'MsgBox "debug 3 -- culvert id: " & pExistingCulvert.OID
    If removeCode > 0 Then
      'set the remove code value into the culvert layer
      'it will be reevaluated in the end
      pExistingCulvert.Value(rmcdIDx) = removeCode
      pExistingCulvert.Store
    Else
      ReDim Preserve ErrorLog(count)
      ErrorLog(count) = pExistingCulvert.OID
      count = count + 1
    End If
    'loop refresh
    Set pExistingCulvert = pCulvCursor.NextFeature
  Loop

  'MsgBox "debug 4"

  'signal that all culverts have been successfuly added
  If count = 0 Then
    ErrorLog(0) = -1
  End If
  SplitRoadsAtExistingCulverts = ErrorLog

  Exit Function
ErrorHandler:
  HandleError False, "SplitRoadsAtExistingCulverts " & c_ModuleFileName
& " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function

'Add one culvert to the network and maintain the topology
'return a code indicating if culvert has split a segment or not
Private Function InsertExistingCulvertLocation(pPointFeat As IFeature, _
```

```
                                    pLineFeatClass As IFeatureClass, _
                                    pSnapAgent As ISnapAgent, _
                                    tolerance As Double, _
                                    RSAFieldInd() As Long) As Integer
  On Error GoTo ErrorHandler

  Dim removeCode As Integer
  removeCode = 0

  'MsgBox "insert culvert debug 1"

  '1 Snap point (culvert) to line (road)
  Util.MovePointFeatToSnapLocation pPointFeat, pSnapAgent, tolerance

  'MsgBox "insert culvert debug 2"

  '3 Break line (road) in two segments at point location (culvert)
  Dim pLineFeature As IFeature
  Set pLineFeature = Util.FindFeatureXPoint(pLineFeatClass,
pPointFeat.Shape)
  If pLineFeature Is Nothing Then Exit Function
  Dim pSegments(2) As IPolyline
  Util.CutPolylineAtPoint pLineFeature.Shape, pPointFeat.Shape,
pSegments

  'MsgBox "insert culvert debug 3"

  '4 Create new features and copy all non topological attributes
  Dim pNewFeat As IFeature
  If (Not pSegments(0) Is Nothing) And (Not pSegments(1) Is Nothing)
Then
    If pSegments(1).Length > 0 And pSegments(0).Length > 0 Then
      Set pNewFeat = pLineFeatClass.CreateFeature
      Util.CopyAllAtributes pLineFeature, pNewFeat
      Set pLineFeature.Shape = pSegments(0) 'upper segment
      Set pNewFeat.Shape = pSegments(1) 'lower segment
    Else
      'CANNOT ADD CULVERT IN INTERSECTION - removing culvert could not
be done without
      'user query in order to rebuild parent-child topology
      'check how many segments this point intersects
      'if more than 2 bail
      Dim pFeatures() As IFeature
      pFeatures =
Util.FindAllFeaturesNearPoint(m_pRoadLayer.FeatureClass, _
                                          pPointFeat.Shape, tolerance)
      If UBound(pFeatures) > 1 Then Exit Function
    End If
  Else
    Exit Function
  End If

  'MsgBox "insert culvert debug 4"

  If Not pNewFeat Is Nothing Then
    '5 Give unique identifiers to new road segments
    Dim maxId As Long
    maxId = Util.GetMaxValue(pLineFeatClass, RSAFieldInd(0))
    If maxId < 0 Then maxId = 0
```

```
    pNewFeat.Value(RSAFieldInd(0)) = maxId + 1
    'signal that the road has sucessfully been split __VERY IMPORTANT__
    removeCode = 1

    'MsgBox "insert culvert debug 5"

    '6 Maintain node topology -- INSERT NODE
    Dim maxPointId As Long
    maxPointId = Util.GetMaxOfFields(pLineFeatClass, RSAFieldInd(3),
RSAFieldInd(4))
    'make parent segment flow to a new node
    pLineFeature.Value(RSAFieldInd(4)) = maxPointId + 1
    'make child flow from the same new node
    pNewFeat.Value(RSAFieldInd(3)) = maxPointId + 1

    'MsgBox "insert culvert debug 6"

    '7 Signal that these features have no parents or child respectively.
    ' This will allow us to keep the node topology intact (see
SetParents)
    ' in order to be able to rebuild the network at removal time.
    pNewFeat.Value(m_lRoadFieldInd(5)) = -1 'Par1
    pNewFeat.Value(m_lRoadFieldInd(6)) = -1 'Par2
    pNewFeat.Value(m_lRoadFieldInd(7)) = -1 'Par3

    'MsgBox "insert culvert debug 7"

    pNewFeat.Store
  Else
    'signal that the culvert has been added to the end of a road segment
__VERY IMPORTANT__
    removeCode = 2
  End If

  'MsgBox "insert culvert debug 8"

  'signal that original feature has no child (see 7)
  pLineFeature.Value(m_lRoadFieldInd(8)) = -1 'Child
  pLineFeature.Store

  InsertExistingCulvertLocation = removeCode

  Exit Function
ErrorHandler:
  HandleError False, "InsertExistingCulvertLocation " & c_ModuleFileName
& " " & GetErrorLineNumberString(Erl), Err.Number, Err.Source,
Err.Description, 4
End Function
'give each culvert an approriate remove code in order to avoid
'abnormalities when user remove culverts later on
Private Function EvaluateCulvertCode() As Boolean
  'the following codes are used:
    '0 - cannot be removed -- stream crossing, road termination
    '1 - general case -- located along a uniform road stretch, segments
will merge at removal
    '2 - end culvert -- located at the edge of a uniform section, no
merge at removal
  Dim lRemIdx As Long, lDelPotIdx As Long
  lRemIdx = m_lCulFieldInd(1)
```

```
  lDelPotIdx = m_lCulFieldInd(2)
  Dim pFeat As IFeature
  Dim pPolyline As IPolyline
  Dim pFeatures() As IFeature
  Dim unremovable As Boolean
  Dim deliverAll As Boolean
  deliverAll = False
  unremovable = False
  Dim i As Integer
  Dim touchCount As Integer
  touchCount = 0

  'select all culverts
'  Dim pFilter As IQueryFilter
'  Set pFilter = New QueryFilter
'  pFilter.WhereClause = "REMCD > 0"
  Dim pCulvCursor As IFeatureCursor
  Set pCulvCursor = m_pCulvertLayer.Search(Nothing, False)

  'Loop through all culverts
  Dim pCurCulv As IFeature
  Set pCurCulv = pCulvCursor.NextFeature
  Do While Not pCurCulv Is Nothing
    'examine if the culvert is on a stream
    Set pFeat = Util.FindFeatureNearPoint(m_pStreamLayer.FeatureClass, _
                                          pCurCulv.Shape, 0.000000001)
    If Not pFeat Is Nothing Then
      'is on a stream
      unremovable = True
      deliverAll = True
    Else
      'culvert is not on a stream
      'examine if the culvert is at the end of one or more segments
      pFeatures =
Util.FindAllFeaturesNearPoint(m_pRoadLayer.FeatureClass, _
                                          pCurCulv.Shape, 0.000000001)
      'count the number of features touching the culvert
      'which flow to this culvert
      For i = 0 To UBound(pFeatures)
        Set pFeat = pFeatures(i)
        If Not pFeat Is Nothing Then
          Set pPolyline = pFeat.Shape
          If Util.ComparePointLocations(pPolyline.ToPoint,
pCurCulv.Shape) Then
            'point is at the TOPT; count this feature
            touchCount = touchCount + 1
          End If
        End If
      Next i

      If (touchCount = 1 And UBound(pFeatures) = 0) Or touchCount > 1
Then
        unremovable = True
      End If
    End If

    If unremovable Then
      'prohibit removing culvert if it is on a stream
      'or on a terminal segment
```

```
       pCurCulv.Value(lRemIdx) = 0
    End If
    If deliverAll Then
      'force the probability of delivery to 1
      'this is to avoid raster inconsistencies when analyzing
      pCurCulv.Value(lDelPotIdx) = 1
    End If

    pCurCulv.Store

    'iterate
    unremovable = False
    deliverAll = False
    touchCount = 0
    Set pFeat = Nothing
    Set pCurCulv = pCulvCursor.NextFeature
  Loop

End Function

Private Sub DisplayErrorReport(success As Boolean)
  If success Then
    Dim response As String
    response = "Network Setup Completed Sucessfuly."
    If m_ErLog(0) > -1 Then
      response = response & vbLf & "The following existing culverts
could not be added"
      Dim i As Integer
      For i = LBound(m_ErLog) To UBound(m_ErLog)
        response = response & m_ErLog(i) & vbLf
      Next i
      response = response & "Try adding them manually at analysis time"
    End If
    MsgBox response
  Else
    MsgBox "Network Setup Completed Unsuccessfuly!", vbCritical
  End If
End Sub
```

CLASS - clsSetUpRoad (clsSetUpRoad.cls)

```
Option Explicit

Implements ICommand

Private m_pApp As IApplication
Private m_pExt As clsExt
Private Const ZFNAME As String = "ZFROM"
Private Const ZTNAME As String = "ZTO"
' Variables used by the Error handler function - DO NOT REMOVE
Const c_ModuleFileName =
"C:\Evenflo\ThesisWorks\VBScripts\CrossDrainSpacer\clsSetUpRoad.cls"
```

```
Private Sub Class_Terminate()
  On Error GoTo ErrorHandler

17:   Set m_pApp = Nothing
18:   Set m_pExt = Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "Class_Terminate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  On Error GoTo ErrorHandler

  'no bitmap needed


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Bitmap " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Caption() As String
  On Error GoTo ErrorHandler

40:   ICommand_Caption = "Road SetUp"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Caption " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Category() As String
  On Error GoTo ErrorHandler

51:   ICommand_Category = "Road Sediment Analyst"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Category " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Checked() As Boolean
  On Error GoTo ErrorHandler

  'TODO: your implementation here
```

```
  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Checked " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Enabled() As Boolean
  On Error GoTo ErrorHandler

73:   If Not m_pExt Is Nothing Then
74:     If m_pExt.IsStarted And Not m_pExt.IsSetUp Then
75:       ICommand_Enabled = True
76:     Else: ICommand_Enabled = False
77:     End If
78:   Else: ICommand_Enabled = False
79:   End If


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Enabled " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpContextID() As Long
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpContextID " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_HelpFile() As String
  On Error GoTo ErrorHandler

  'TODO: your implementation here


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_HelpFile " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Message() As String
  On Error GoTo ErrorHandler

112:   ICommand_Message = "Set Up The Required Road/Ditch Structure"


  Exit Property
```

```
ErrorHandler:
  HandleError True, "ICommand_Message " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Property Get ICommand_Name() As String
  On Error GoTo ErrorHandler

123:   ICommand_Name = "SetUp"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Name " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Sub ICommand_OnClick()
  On Error GoTo ErrorHandler


  'check exension
136:   If m_pExt Is Nothing Then
137:     MsgBox "invalid RSA extension"
    Exit Sub
139:   End If
  'get layer
  Dim pRoadLayer As IFeatureLayer
142:   Set pRoadLayer = m_pExt.RoadLayer
  'get dem
  Dim pDemLayer As IRasterLayer
145:   Set pDemLayer = m_pExt.DemLayer
  'check editing state
  Dim pWksEdit As IWorkspaceEdit
  Dim pDS As IDataset
149:   Set pDS = pRoadLayer
150:   Set pWksEdit = pDS.Workspace
151:   If pWksEdit.IsBeingEdited Then
152:     MsgBox "The workspace: " & pDS.Workspace.PathName & vbLf & _
           "is currently being edited. Stop editing first"
    Exit Sub
155:   End If
  'get grade name from user
  Dim pFrm As frmGradeName
158:   Set pFrm = New frmGradeName
159:   pFrm.RSAExtension = m_pExt
160:   pFrm.Show vbModal
  Dim ok As Boolean, bOverride As Boolean
162:   ok = pFrm.CompletedOK
163:   bOverride = pFrm.OverrideValues
164:   Set pFrm = Nothing
  If Not ok Then Exit Sub
  'create fields
  If Not CreateZFields(pRoadLayer.FeatureClass) Then Exit Sub
  'set cursor to busy
  Dim pMouseCur As IMouseCursor
170:   Set pMouseCur = New MouseCursor
```

```
171:   pMouseCur.SetCursor 2
  'start editing
173:   pWksEdit.StartEditing False
  'enforce simple paths
175:   EnforceFnct.SimplifyPaths pRoadLayer.FeatureClass
  'enforce end connectivity only, no midway intersections
177:   EnforceFnct.ForceEndConnectivity pRoadLayer.FeatureClass
  'sample z values from DEM
179:   GetEndPointsElevation pRoadLayer, pDemLayer
  'set flow directionality
181:   SetFlowAlongSegments pRoadLayer
  'compute grade
183:   If bOverride Then EstimateGrade pRoadLayer.FeatureClass
   'save changes
185:   pWksEdit.StopEditing True
  'reset cursor
187:   pMouseCur.SetCursor 0
  'display layer with arrows
189:   DisplayWithArrows pRoadLayer
  'refresh
  Dim pDoc As IMxDocument
192:   Set pDoc = m_pApp.Document
193:   pDoc.ActiveView.PartialRefresh esriViewGeography +
esriViewGraphics, pRoadLayer, Nothing
  'signal success and turn on buttons
195:   m_pExt.IsSetUp = True
  'release memory
197:   Set pDoc = Nothing
198:   Set pMouseCur = Nothing
199:   Set pWksEdit = Nothing
200:   Set pRoadLayer = Nothing
201:   Set pDemLayer = Nothing


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnClick " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  On Error GoTo ErrorHandler

212:   Set m_pApp = hook
  Dim pId As New UID
214:   pId.Value = "RoadSedimentAnalyst.clsExt"
215:   Set m_pExt = m_pApp.FindExtensionByCLSID(pId)


  Exit Sub
ErrorHandler:
  HandleError True, "ICommand_OnCreate " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Property Get ICommand_Tooltip() As String
  On Error GoTo ErrorHandler
```

```
226:   ICommand_Tooltip = "Set Up The Required Ditch Structure"


  Exit Property
ErrorHandler:
  HandleError True, "ICommand_Tooltip " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Property

Private Sub DisplayWithArrows(pLayer As ILayer)
  On Error GoTo ErrorHandler


  Dim pGFLayer As IGeoFeatureLayer
  Dim pSimpleRenderer As ISimpleRenderer
  Dim pNewSymbol As ILineSymbol
  Dim pColor As IColor
  ' Check if the layer is a feature layer
  If Not TypeOf pLayer Is IGeoFeatureLayer Then Exit Sub
244:   Set pGFLayer = pLayer
  ' Check if there is a simple renderer and get a reference to it
  If Not TypeOf pGFLayer.Renderer Is ISimpleRenderer Then Exit Sub
247:   Set pSimpleRenderer = pGFLayer.Renderer
248:   Set pNewSymbol = pSimpleRenderer.Symbol
  'get initial color
250:   Set pColor = pNewSymbol.Color
  'create new line symbol
252:   Set pNewSymbol = New CartographicLineSymbol
253:   pNewSymbol.Color = pColor
254:   pNewSymbol.Width = 1
  'create arrow
  Dim pLineProps As ILineProperties
257:   Set pLineProps = pNewSymbol
  Dim pLinedec As ILineDecoration
259:   Set pLinedec = New LineDecoration
  Dim pLDE As ISimpleLineDecorationElement
261:   Set pLDE = New SimpleLineDecorationElement
  Dim pArrow As IArrowMarkerSymbol
263:   Set pArrow = New ArrowMarkerSymbol
264:   With pArrow
265:     .Color = pColor
266:     .Size = 7
    '.Width = 6
268:   End With
  'add arrow to line
270:   pLDE.MarkerSymbol = pArrow
271:   pLDE.AddPosition 1
272:   pLinedec.AddElement pLDE
273:   Set pLineProps.LineDecoration = pLinedec
  'add new symbol to rendered
275:   Set pSimpleRenderer.Symbol = pNewSymbol



  Exit Sub
ErrorHandler:
```

```
  HandleError False, "DisplayWithArrows " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Sub

Private Function CreateZFields(pRoadClass As IFeatureClass) As Boolean
  On Error GoTo ErrorHandler


  Dim pField As IField
  Dim pFieldEdit As IFieldEdit

291:   Set pField = New Field
292:   Set pFieldEdit = pField

294:   With pFieldEdit
   'fields of type double
296:     .Name = ZFNAME
297:     .Type = esriFieldTypeDouble
298:     Util.AddNonExistingField pField, pRoadClass

300:     .Name = ZTNAME
301:     Util.AddNonExistingField pField, pRoadClass
302:   End With

304:   CreateZFields = True

306:   Set pField = Nothing
307:   Set pFieldEdit = Nothing


  Exit Function
ErrorHandler:
312:   CreateZFields = False
  HandleError False, "CreateZFields " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function
'has to run inside an edit session.
Private Function GetEndPointsElevation(pRoadLayer As IFeatureLayer,
pDemLayer As IRasterLayer) As Boolean
  On Error GoTo ErrorHandler


  '********************************************** get vector stuff
**************************
  'Get road selection
  Dim pSelectionSet As ISelectionSet
  Dim pRoadSelection As IFeatureSelection
324:   Set pRoadSelection = pRoadLayer
  Dim pRoadCursor As IFeatureCursor

  'MsgBox "check 1"

329:   Set pSelectionSet = pRoadSelection.SelectionSet
  'if no selection select all
331:   If pSelectionSet.count = 0 Then
332:     Set pRoadCursor = pRoadLayer.Search(Nothing, False)
333:   Else
```

```
334:     pSelectionSet.Search Nothing, False, pRoadCursor
335:   End If

  'MsgBox "check 2"

  Dim pRoad As IFeature
  Dim pPolyline As IPolyline
341:   Set pRoad = pRoadCursor.NextFeature

   'get required field indexes
  Dim lZFIndex As Long, lZTIndex As Long
345:   lZFIndex = pRoadLayer.FeatureClass.FindField(ZFNAME)
346:   lZTIndex = pRoadLayer.FeatureClass.FindField(ZTNAME)

  'MsgBox "check 3"

  '********************************************** Loop and get values
**********************
  Dim sValue As String
352:   Do While Not pRoad Is Nothing
353:     Set pPolyline = pRoad.Shape
    'get value of the from point
355:     sValue = Util.GetCellValue(pDemLayer, pPolyline.FromPoint)
356:     If StrComp(sValue, "NoData", vbTextCompare) <> 0 Then
357:       pRoad.Value(lZFIndex) = CDbl(sValue)
358:     End If
    'get value of the to point
360:     sValue = Util.GetCellValue(pDemLayer, pPolyline.ToPoint)
361:     If StrComp(sValue, "NoData", vbTextCompare) <> 0 Then
362:       pRoad.Value(lZTIndex) = CDbl(sValue)
363:     End If
364:     pRoad.Store
    'increase count
366:     Set pRoad = pRoadCursor.NextFeature
367:   Loop

  'MsgBox "check 4"
370:   GetEndPointsElevation = True


  Exit Function
ErrorHandler:
375:  GetEndPointsElevation = False
  HandleError False, "GetEndPointsElevation " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function
'has to run inside edit session
Private Function SetFlowAlongSegments(pRoadLayer As IFeatureLayer) As
Boolean
  On Error GoTo ErrorHandler

  Dim pRoadSeg As IFeature

  'select all features in pRoadLayer
  Dim pFeatCur As IFeatureCursor
388:   Set pFeatCur = pRoadLayer.Search(Nothing, False)
389:   Set pRoadSeg = pFeatCur.NextFeature
   'get required field indexes
```

```
  Dim lZFIdx As Long, lZTIdx As Long
392:   lZFIdx = pRoadLayer.FeatureClass.FindField(ZFNAME)
393:   lZTIdx = pRoadLayer.FeatureClass.FindField(ZTNAME)

  'go through each segment, verify flow and change if necessary
  Dim dZFrom As Double, dZTo As Double
  Dim pPolyline As IPolyline
398:   Do While Not pRoadSeg Is Nothing
399:     dZFrom = pRoadSeg.Value(lZFIdx)
400:     dZTo = pRoadSeg.Value(lZTIdx)
401:     If (dZFrom < dZTo) Then
402:       Set pPolyline = pRoadSeg.Shape
403:       pPolyline.ReverseOrientation
404:       Set pRoadSeg.Shape = pPolyline
405:       Debug.Print "flipped " & pRoadSeg.OID
      'swap values in table
407:       pRoadSeg.Value(lZFIdx) = dZTo
408:       pRoadSeg.Value(lZTIdx) = dZFrom

410:       pRoadSeg.Store

412:     End If

414:     Set pRoadSeg = pFeatCur.NextFeature
415:   Loop

417:   SetFlowAlongSegments = True


  Exit Function
ErrorHandler:
422:   SetFlowAlongSegments = False
  HandleError False, "SetFlowAlongSegments " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function
'run inside edit session
Private Function EstimateGrade(pFClass As IFeatureClass) As Boolean
  On Error GoTo ErrorHandler


  Dim grade As Integer
  Dim d As Double, L As Double, h As Double
  'get field indexes
  Dim lZFi As Long, lZTi As Long, lGi As Long
434:   lZFi = pFClass.FindField(ZFNAME)
435:   lZTi = pFClass.FindField(ZTNAME)
436:   lGi = pFClass.FindField(m_pExt.GradeName)
  'loop
  Dim pCursor As IFeatureCursor
439:   Set pCursor = pFClass.Search(Nothing, False)
  Dim pFeat As IFeature
441:   Set pFeat = pCursor.NextFeature
  Dim pPolyline As IPolyline
443:   Do While Not pFeat Is Nothing
444:     Set pPolyline = pFeat.Shape
445:     L = pPolyline.Length
446:     h = Math.Abs(pFeat.Value(lZFi) - pFeat.Value(lZTi))
447:     d = Math.Sqr(Math.Abs(L * L - h * h))
```

```
448:     grade = CInt(h * 100 / d)
449:     pFeat.Value(lGi) = grade
450:     pFeat.Store
451:     Set pFeat = pCursor.NextFeature
452:   Loop

  'relese memory
455:   Set pCursor = Nothing
456:   Set pFeat = Nothing
457:   Set pPolyline = Nothing



  Exit Function
ErrorHandler:
  HandleError False, "EstimateGrade " & c_ModuleFileName & " " &
GetErrorLineNumberString(Erl), Err.Number, Err.Source, Err.Description,
4
End Function
```

CLASS - clsSplit (clsSplit.cls)

```
Option Explicit

Implements ICommand
Implements ITool

Private m_pApp As IApplication
Private m_pBitmap As IPictureDisp
Private m_pMouseCur As IPictureDisp
Private m_pExt As clsExt
Private m_pFeatClass As IFeatureClass
Private m_pWksEdit As IWorkspaceEdit
Private m_pRefresh As IInvalidArea

Private m_pDisplay As IDisplay
Private m_pSymbol As ISymbol
Private m_pNewPoint As IPoint
Private m_pSnapAgent As IFeatureSnapAgent

Private Sub Class_Initialize()
  'load the button image from the resource file
  Set m_pBitmap = LoadResPicture("Split", vbResBitmap)
  Set m_pMouseCur = LoadResPicture("EditLine", vbResCursor)
End Sub

Private Sub Class_Terminate()
  Set m_pBitmap = Nothing
  Set m_pMouseCur = Nothing
  Set m_pExt = Nothing
  Set m_pApp = Nothing
End Sub

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  ICommand_Bitmap = m_pBitmap
End Property
```

```
Private Property Get ICommand_Caption() As String
  ICommand_Caption = "Split"
End Property

Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean
  'TODO: your implementation here
End Property

Private Property Get ICommand_Enabled() As Boolean
  'check for certain properties in extension
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted And m_pExt.IsSetUp And Not m_pExt.HasTopology
Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long
  'TODO: your implementation here
End Property

Private Property Get ICommand_HelpFile() As String
  'TODO: your implementation here
End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Split Segment At Point"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "Split"
End Property

Private Sub ICommand_OnClick()
  On Error GoTo erh
  Dim pMxDoc As IMxDocument
  Set pMxDoc = m_pApp.Document
  Set m_pDisplay = pMxDoc.ActiveView.ScreenDisplay
  Set m_pFeatClass = m_pExt.RoadLayer.FeatureClass
  'create new snap agent
  Set m_pSnapAgent = New FeatureSnap
  With m_pSnapAgent
    Set .FeatureClass = m_pFeatClass
    .HitType = esriGeometryPartBoundary
  End With
  'create new symbol
  Set m_pSymbol = New SimpleMarkerSymbol
  m_pSymbol.ROP2 = esriROPNotXOrPen
  Dim pMarkSym As IMarkerSymbol
  Set pMarkSym = m_pSymbol 'QI
  Dim myColor As IColor
```

```
  Set myColor = New RgbColor
  myColor.RGB = RGB(0, 0, 0)
  pMarkSym.Color = myColor
  pMarkSym.Size = 8
  'get the workspace to edit
  Dim pDS As IDataset
  Set pDS = m_pFeatClass
  Set m_pWksEdit = pDS.Workspace
  If Not m_pWksEdit.IsBeingEdited Then
    m_pWksEdit.StartEditing True
  End If
  'create new screen refresh
  Set m_pRefresh = New InvalidArea
  Set m_pRefresh.Display = m_pDisplay

  Exit Sub
erh:
  MsgBox "error in create flip cmd: " & Error
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String
  ICommand_Tooltip = "Split Segment At Point"
End Property

Private Property Get ITool_Cursor() As esriCore.OLE_HANDLE
  ITool_Cursor = m_pMouseCur
End Property

Private Function ITool_Deactivate() As Boolean
  DrawSymbol m_pNewPoint
  Set m_pNewPoint = Nothing 'this will avoid marker leftovers
  Set m_pDisplay = Nothing
  Set m_pSymbol = Nothing

  If Not m_pWksEdit Is Nothing Then
    m_pWksEdit.StopEditing True
  End If
  Set m_pWksEdit = Nothing
  Set m_pRefresh = Nothing

  ITool_Deactivate = True
End Function

Private Function ITool_OnContextMenu(ByVal X As Long, ByVal Y As Long)
As Boolean
  'TODO: your implementation here
End Function

Private Sub ITool_OnDblClick()
  'unused
End Sub
```

```
Private Sub ITool_OnKeyDown(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnKeyUp(ByVal keyCode As Long, ByVal Shift As Long)
  'TODO: your implementation here
End Sub

Private Sub ITool_OnMouseDown(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  If Not m_pNewPoint Is Nothing Then
    Dim pFeat As IFeature
    Dim pNewFeat As IFeature
    Set pFeat = Util.FindFeatureXPoint(m_pFeatClass, m_pNewPoint)
    'get the feature intersected by the point
    If Not pFeat Is Nothing Then
      m_pWksEdit.StartEditOperation
      'split old geometry
      Dim pSegments(1) As IPolyline
      Util.CutPolylineAtPoint pFeat.Shape, m_pNewPoint, pSegments
      If (Not pSegments(0) Is Nothing) And (Not pSegments(1) Is Nothing)
Then
        'create new feature
        Set pNewFeat = m_pFeatClass.CreateFeature
        'copy attributes
        Util.CopyAllAtributes pFeat, pNewFeat
        'set geometries
        Set pFeat.Shape = pSegments(0)
        Set pNewFeat.Shape = pSegments(1)

        pFeat.Store
        pNewFeat.Store
        m_pRefresh.Add pFeat
        m_pRefresh.Add pNewFeat
      End If
      m_pWksEdit.StopEditOperation
      m_pRefresh.Invalidate esriAllScreenCaches
    End If
  End If
End Sub

Private Sub ITool_OnMouseMove(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  DrawSymbol m_pNewPoint
  Set m_pNewPoint = m_pDisplay.DisplayTransformation.ToMapPoint(X, Y)
  DrawSymbol m_pNewPoint
End Sub

Private Sub ITool_OnMouseUp(ByVal Button As Long, ByVal Shift As Long,
ByVal X As Long, ByVal Y As Long)
  'not used
End Sub

Private Sub ITool_Refresh(ByVal hDC As esriCore.OLE_HANDLE)
  'avoid a marker left on the line
  Set m_pNewPoint = Nothing
End Sub

Sub DrawSymbol(pPoint)
```

```
  On Error GoTo erh
  If Not pPoint Is Nothing Then 'the point is initialy nothing
    m_pDisplay.StartDrawing m_pDisplay.hDC, esriNoScreenCache
    m_pSymbol.SetupDC m_pDisplay.hDC, m_pDisplay.DisplayTransformation
    m_pSnapAgent.Snap Nothing, pPoint, 100
    m_pSymbol.Draw pPoint
    m_pSymbol.ResetDC
    m_pDisplay.FinishDrawing
  End If
  Exit Sub
erh:
  MsgBox "error in draw symbol " & Error
End Sub
```

CLASS – clsStart (clsStart.cls)

```
Option Explicit

Implements ICommand
Private m_pApp As IApplication
Private m_pDoc As IMxDocument
Private m_pExtension As clsExt

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  'TODO: your implementation here
End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "Start"
End Property

Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property

Private Property Get ICommand_Checked() As Boolean
  'TODO: your implementation here
End Property

Private Property Get ICommand_Enabled() As Boolean
  If Not m_pExtension Is Nothing Then
    If Not m_pExtension.IsStarted Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long
  'TODO: your implementation here
End Property

Private Property Get ICommand_HelpFile() As String
  'TODO: your implementation here
End Property
```

```
Private Property Get ICommand_Message() As String
  ICommand_Message = "Starts the Road Sediment Analyst"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "Start"
End Property

Private Sub ICommand_OnClick()
  'disable buttons
  m_pExtension.IsStarted = False
  m_pExtension.IsSetUp = False

  'Verify that sediment model exists
  Dim pSedModel As ISedimentModel
  Set pSedModel = m_pApp.FindExtensionByName(m_pExtension.SedModelName)
  If pSedModel Is Nothing Then
    MsgBox "No Sediment Model could be found!" & vbLf & "Please check
extesions."
    Exit Sub
  ElseIf TypeOf pSedModel Is IExtensionConfig Then
    Dim pExtConfig As IExtensionConfig
    Set pExtConfig = pSedModel
    If pExtConfig.State = esriESDisabled Then
      pExtConfig.State = esriESEnabled
    End If
    Set pExtConfig = Nothing
  End If

  'get the layers form user and set them up in the extension
  Dim pWorkSpace As IWorkspace
'  Set pWorkSpace = pCulvLayer.FeatureClass.FeatureDataset.Workspace
  Dim pFrm As frmSetData
  Set pFrm = New frmSetData
  pFrm.AddLayers m_pDoc.FocusMap, Me
  pFrm.Show vbModal
  'after form has released the following will happen
  If pFrm.CompletedSuccessfuly = True Then
  'set extension started
  m_pExtension.IsStarted = True
  End If
  'destroy form reference
  Set pFrm = Nothing
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  Set m_pApp = hook
  Set m_pDoc = m_pApp.Document
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExtension = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String
  ICommand_Tooltip = "Starts the Road Sediment Analyst"
End Property
'variable iCopy can be 0 > no copies or 1 > make copies
Public Sub ReceiveLayers(sRLName As String, sSLName As String, sCLName
As String, _
```

```
                        sDEMName As String, iCopy As Integer)
  If m_pExtension Is Nothing Then
    MsgBox "could not find the road analyst extension"
    Exit Sub
  End If

  'identify all datasets by name
  Dim pRoadLayer As IFeatureLayer
  Dim pStreamLayer As IFeatureLayer
  Dim pCulvertLayer As IFeatureLayer
  Dim pDemLayer As IRasterLayer

  Set pRoadLayer = GetLayer(sRLName)
  Set pStreamLayer = GetLayer(sSLName)
  Set pCulvertLayer = GetLayer(sCLName)
  Set pDemLayer = GetLayer(sDEMName)

  If iCopy = 1 Then
    'make copies of the datasets
  End If

  'set these datasets into the extension
  m_pExtension.RoadLayer = pRoadLayer
  m_pExtension.StreamLayer = pStreamLayer
  m_pExtension.CulvLayer = pCulvertLayer
  m_pExtension.DemLayer = pDemLayer

End Sub

Private Function GetLayer(sName As String) As IDataLayer
  Dim counter As Integer
  For counter = 0 To m_pDoc.FocusMap.LayerCount - 1
    If sName = m_pDoc.FocusMap.Layer(counter).Name Then
      Set GetLayer = m_pDoc.FocusMap.Layer(counter)
      Exit Function
    End If
  Next counter
End Function
```

CLASS - clsStop (clsStop.cls)

```
Option Explicit

Implements ICommand
Private m_pApp As IApplication
Private m_pExt As clsExt

Private Property Get ICommand_Bitmap() As esriCore.OLE_HANDLE
  'TODO: your code here
End Property

Private Property Get ICommand_Caption() As String
  ICommand_Caption = "Stop"
End Property

Private Property Get ICommand_Category() As String
  ICommand_Category = "Road Sediment Analyst"
End Property
```

```
Private Property Get ICommand_Checked() As Boolean
  'TODO: your code here
End Property

Private Property Get ICommand_Enabled() As Boolean
  If Not m_pExt Is Nothing Then
    If m_pExt.IsStarted Then
      ICommand_Enabled = True
    Else: ICommand_Enabled = False
    End If
  Else: ICommand_Enabled = False
  End If
End Property

Private Property Get ICommand_HelpContextID() As Long
  'TODO: your code here
End Property

Private Property Get ICommand_HelpFile() As String
  'TODO: your code here
End Property

Private Property Get ICommand_Message() As String
  ICommand_Message = "Stop Cross Drain Analyst"
End Property

Private Property Get ICommand_Name() As String
  ICommand_Name = "Stop"
End Property

Private Sub ICommand_OnClick()
  m_pExt.IsStarted = False
  m_pExt.IsSetUp = False
  m_pExt.HasTopology = False
  m_pExt.IsAnalyzed = False
  m_pExt.RoadLayer = Nothing
  m_pExt.StreamLayer = Nothing
  m_pExt.CulvLayer = Nothing
  m_pExt.DemLayer = Nothing

  Dim pSedModel As ISedimentModel
  Set pSedModel = m_pApp.FindExtensionByName(m_pExt.SedModelName)
  pSedModel.StopSession

   m_pExt.TriggerStopEvent
End Sub

Private Sub ICommand_OnCreate(ByVal hook As Object)
  Set m_pApp = hook
  Dim pId As New UID
  pId.Value = "RoadSedimentAnalyst.clsExt"
  Set m_pExt = m_pApp.FindExtensionByCLSID(pId)
End Sub

Private Property Get ICommand_Tooltip() As String

End Property
```

CLASS - clsToolBar (clsToolBar.cls)

```
Option Explicit

Implements IToolBarDef

Private Property Get IToolBarDef_Caption() As String
  IToolBarDef_Caption = "Road Sediment Analyst"
End Property

Private Sub IToolBarDef_GetItemInfo(ByVal pos As Long, ByVal itemDef As
esriCore.IItemDef)
  Dim pUID As New UID
  itemDef.Group = False

  Select Case pos
    Case 0
      pUID.Value = "RoadSedimentAnalyst.clsMenu"
    Case 1
      itemDef.Group = True
      pUID.Value = "RoadSedimentAnalyst.clsCrtCulv"
    Case 2
      pUID.Value = "RoadSedimentAnalyst.clsMoveCulv"
    Case 3
      pUID.Value = "RoadSedimentAnalyst.clsRmvCulv"
    Case 4
      pUID.Value = "RoadSedimentAnalyst.clsSedBox"
    Case 5
      itemDef.Group = True
      pUID.Value = "RoadSedimentAnalyst.clsSegGrade"
    Case 6
      pUID.Value = "RoadSedimentAnalyst.clsNodeGrade"
    Case 7
      itemDef.Group = True
      pUID.Value = "RoadSedimentAnalyst.clsFlip"
    Case 8
      pUID.Value = "RoadSedimentAnalyst.clsSplit"
    Case 9
      pUID.Value = "RoadSedimentAnalyst.clsMerge"
    Case 10
      itemDef.Group = True
      pUID.Value = "RoadSedimentAnalyst.clsEnforCon"
  End Select

  itemDef.ID = pUID
End Sub

Private Property Get IToolBarDef_ItemCount() As Long
  IToolBarDef_ItemCount = 11
End Property

Private Property Get IToolBarDef_Name() As String
  IToolBarDef_Name = "Road Sediment Analyst Toolbar"
End Property
```