



Training in the use of Gedae.



# Gedae Introductory Course Syllabus

The Gedae Introductory Course consists of 16 lectures and most with a related laboratory. We cover a wide range of topics that will provide a solid foundation for using Gedae to develop applications.

## TOPICS:

### 1. Essentials of Data Flow

This lecture introduces Gedae and the concept of data flow. The impact of Gedae on development methodologies is discussed briefly. The structure of Gedae's block diagram language and internal software is explained to give you a context for the material that follows. The general theory of data flow is explained to provide a foundation for understanding Gedae's data flow. During the laboratory the student will develop and deploy an audio processing application.

### 2. Development Methodology

This lecture introduces the general methodologies that you can apply when using Gedae to develop software applications. While the process is generally the same as traditional code development processes – develop algorithms, define system requirements, develop application specification, implement algorithms, integrate and test – there are several details that you can do differently to provide better results and productivity.

### 3. Static Data Flow

This lecture introduces Gedae's concept of static data flow. Data flow is data port characteristics and associated processing methods. With static data flow the port characteristics do not change during the course of execution so that execution can be fully scheduled during the compilation of the application. We will define the terms that define static characteristics of function box input and output ports.

### 4. Static Data Flow Primitives – Parts I & II

These lectures introduce the structure and syntax of Gedae's primitives. In order to effectively use Gedae, you must be able to analyze primitives to determine if they have the behavior you need. You must also be able to create your own custom primitives. This lecture draws a direct relationship between the static data flow concepts defined in the previous lecture and the syntax and semantics of Gedae's static data flow language features. It also describes some of functions provided by Gedae for information exchange with Gedae's runtime kernel.

### 5. Example: Generating a Synthetic Signal

This lecture illustrates the use of static dataflow by generating a synthetic RADAR signal that will be used to drive the detection algorithm used in the next lecture. In the laboratory for this lecture you will enhance the source so that it is moving along a predefined path. You must also create the signal as the RADAR sensor scans. You will create a simple channel model that adds Gaussian white noise to the signal. You will be provided with all equations that need to be implemented. It is not necessary to understand the underlying RADAR theory.



## 6. C Structures as Gedae Data Elements

This lecture describes the declaration of C structures as data elements in Gedae primitives. The data elements can be any C data structure. The lecture illustrates C structures with a data structure containing the information necessary to define a detection (potential target) in the noisy signal. In the next lecture we will make the output of our detection box dynamic and feed those contacts into a tracking algorithm.

## 7. Dynamic Data Flow

This lecture defines data flow where the amount of data consumed and produced by a function box I/O ports vary at runtime. The lecture also defines the syntax of data flow primitives with dynamic inputs or outputs. The example used to illustrate the idea will be a box that produces tokens only when a potential target is found in the signal. During the laboratory you will explore the structure and behavior of the graph exhibiting dynamic behavior. During this laboratory you will create the detection primitive using a dynamic output.

## 8. Variable Threshold Data Flow

This lecture defines data flow where the threshold of a function box I/O port varies at runtime. During the laboratory you will create a graph that switches between 2 processing modes. You will also study the sensor simulation graph that generates an appropriate signal and the graph that executes the second mode of operation. In a later lecture we will see how segmentation can be used to reset the mode software before the start of each new mode. (I need a second mode from John.)

## 9. Graph Syntax and Semantics – Explicit and Implicit Parameter Types

This lecture introduces the syntax of the graphical editor. We begin by defining the block diagram syntax. Parameters that change asynchronously with data flow are defined in detail. They are broken into three implicit groups – runtime, data flow and instantiation. These are new terms that we define carefully. During the laboratory you will build a graph that illustrates all types of parameters and explore the behavior.

## 10. Graph Syntax and Semantics – Families and Route Boxes

This lecture continues the definition of graph syntax and semantics by defining the family notation. Family notation allows for defining multiple copies of boxes or data items. The syntax introduces a need for route boxes that provide for complex patterns of connections between family members. An example in the lecture parallel FFT processing using a round robin distribution scheme. During the laboratory for this session you will develop a simulation of multiple RADAR reflectors where each family member simulates the signal for one reflector.

## 11. Segmented Data Flow

This lecture continues the development of the data flow language by defining the concept of data stream segments. Data stream segments marks boundaries in an otherwise infinite stream of data. On these boundaries the software is reset and special boundary processing can be completed. During this laboratory you will convert the 2 mode graph into a segmented 2 mode graph and explore the changes in behavior.



## 12. Synchronizing Parameter Changes with Data Flow

This lecture continues the definition of graph syntax and semantics illustrating 2 techniques for synchronizing parameter changes. The first technique uses segmentation and the second technique introduces the concept of run length encoded data streams. During the laboratory for this lecture you will explore the behavior of a graph using run length encoded streams for parameter synchronization. You will also build a graph that uses segmentation to synchronize a parameter change.

## 13. How Gedae Implements Static Data Flow

This lecture introduces the concepts of implementing a functional block diagram by Gedae. It describes how the block diagram is separated into execution threads and how the user can further partition the functionality for mapping to multiple processors. We will use the RADAR processing examples developed in earlier lectures to illustrate these concepts. During the laboratory you will partition a graph and observe the changes in the execution threads generated.

## 14. How Gedae Implements Dynamic, Variable Threshold and Segmented Data Flow

This lecture introduces the concepts of Gedae creating multiple threads of execution to implement the variable behavior boxes with dynamic, variable threshold or segmented inputs or outputs. During the laboratory you will explore implementation of an iterative algorithm using dynamic and variable threshold data flow.

## 15. Summary of the Gedae Implementation Specification GUIs

This lecture describes each of the major components of the Group Control dialog, which is the center for specification of implementation by the user.

## 16. Optimizing Memory

This lecture illustrates by example the methods used to improve the use of resources in a multiprocessor system. It describes each of the major components of the Group Control dialog, which is the center for specification of implementation by the user.

## 17. Optimizing Execution Efficiency by Increasing Granularity

This lecture describes the efficiencies that can be attained by increasing the processing granularity of an execution thread. There is a short lecture followed by a laboratory where you measure the throughput as a function of granularity. You will be able to compare the effect on PCs, PPCs and TigerSHARCs.

## 18. Optimizing Execution by Distributing Processing Across Multiple Processors

This lecture illustrates the increase in throughput that can be achieved by implementing the parallel processing of RADAR data sets. The lecture will describe the techniques. During the laboratory you will implement and explore the parallel processing of both range and azimuth processing for SAR. You will be able to compare the optimization and performance on PCs, PPCs and TigerSHARCs.



### 19. **Optimizing Execution by Subscheduling**

This lecture will define the technique of subscheduling and illustrate the effect with a SONAR processing example. During the laboratory you will explore the effect of subscheduling while running the SAR application on Mercury and Champ AV quad PPC boards.

### 20. **Gedae's E and Function Box Library**

This lecture consists of a review of a table of available primitives and E functions (Gedae's vector library) and a demonstration of the search tool used to locate a primitive among the 7000+ primitives in Gedae's core library. We will also define the naming convention that we will be rolling out later this year.

### 21. **Building GUI's**

This lecture introduces eval and trigger primitives for processing parameters and the GUI library provided by Gedae. During the laboratory you will create a GUI for displaying data before and after filtering.

### 22. **Graph Control**

This lecture describes techniques for controlling the execution of a graph. The lecture will illustrate techniques for using montecarlo experiments to collect statistical behavior of an application. The example will also show how calculate and display a family of curves for various SNRs that plot the probability of detection as a function of the false alarm rate. During the laboratory the student will explore that graph and add control components to the graph built during the previous lecture to manually analyze the effect of parameter changes on graph functionality.

### 23. **Building a Standalone (Deployable) Application**

Create a command program that allows the user to select from a number of windows by using a command line argument.

### 24. **Programming the Cell Broadband Engine Processor**

This lecture will illustrate how some of the concepts and features discussed previously in the course can be applied when using Gedae to program the Cell/B.E. architecture. The lecture will demonstrate how the memory footprint of a large data set can be reduced without changing the flow graph so the application can fit in the SPE cores' small local storage. The lecture will also introduce several advanced techniques for using and optimizing for the Cell/B.E. processor, including utilizing the system memory in the processor to transfer data. Two Playstation 3s and a Cell/B.E. blade server will be available for the hands-on tutorial.

### 25. **Gedae's MATLAB Library**

This lecture discusses the use of Gedae's MATLAB library.