Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

CREATING SIMULATION RUNTIMES FOR PROFIT

Cliff B. King
F&H Simulations, Inc.
P.O. Box 658
Orem, Utah 84059-0658

ABSTRACT

The creation of runtimes in the simulation industry is fast becoming a profitable venture for many business consultants. The majority of small companies lack the expertise and the resources to build their own simulation model. That's were the consultant comes in. The consultant develops a runtime custom made to meet the needs of the client. Now the client has a powerful tool that they can use as part of their ongoing decision making.

The consultant who completes a simulation project, shows the results, gives a few suggestions, and then leaves; deprives the client of the real benefits of simulation; that is, dynamic decision making. On the other hand, the consultant who, in addition to providing the usual consulting services, develops a simulation runtime for the client to use as future decisions have to be made, greatly enhances the overall consulting service. As the effort required to generate simulation runtimes decreases, more consultants will start using them in their practice, and more people will begin using simulation in their business. With proper training, and a well-built simulation runtime, everyone from plant managers to accountants can take advantage of simulation for what it is -- a decision making tool.

1 INTRODUCTION

Taylor II Simulation software allows the interactive creation of custom runtimes. The runtime can easily be set up within the Taylor II menu system, and then saved as a separate executable program. The runtime can be developed to have a defined menu structure all of its own, or at any point in the application, branch into the standard menu structure of Taylor II. Custom input screens particular to the industry or company targeted can be created to prompt the user for the exact information needed (i.e. number of AGV's?, speed of conveyors, batch size for an oven). Predefined reports and graphs can be set up for easy access through a single menu selection.

2 APPROACH

The runtime development kit is integrated in Taylor II, but after you have developed and tested the application, you can run the application as a separate stand alone program. Runtime applications are developed using hierarchically structured sequences of actions. In other words, you define the action that will occur, the level it will occur at, and the level the application will go to when the action is complete.

The following is a conceptual example of what is meant by 'hierarchically structured sequences'. This is an example of a possible runtime application to help with weekly staffing decisions at a bank planning to offer a flexible work schedule for their employees.

Level 1. Main Menu:
Modify/Run/Results/Exit
Level 11. Modify Menu: *Arrival Rates/Tellers/Transactions/Service Times*
Level 111. Arrival Rates: *Define/Edit*
Level 112. Tellers: *Choose Employees/ Define Schedules*
Level 113. Transactions: *Define/Edit*
Level 114. Service Times: *Define/Edit*
Level 12. Run Menu: *No. of Iterations/Length of Simulation*
Level 13. Results Menu: *Reports/Graphs*
Level 14. Run Menu: *Save/Quit*

Notice how the application starts at level 1, and depending on the menu item chosen, the application may advance to levels 11, 12, 13, and 14. If *Modify* is chosen, the application advances to level 11, if *Tellers* is chosen next, then the application advances to level 112. Of course the proper Taylor commands must be defined at each level in order to get the action desired from the chosen menu item. This will be explained in detail now.

To develop a runtime in Taylor II, you first select <Options> from the main menu, and then select <Runkit>. This will bring up three menu selections to choose from:

<Define>	To develop the runtime application
<Test>	To run and test the application while in Taylor II
<File>	To save or retrieve an application (runtimes are saved separately from models).

After selecting <Define>, a window pops up in which you can develop your runtime application. The window contains a table of three columns and multiple rows. Each row represents a level in the runtime application. The three columns are labeled: Level, Action, and Parameters. At each row you will define the current level number, the number of the level to go to when the action is complete, the action itself, and any parameters associated with the action. By hitting <Enter> when the Action field is highlighted, you are presented with a list of available actions to choose from. The type of action chosen will determine what parameters must be entered. For instance, if *Menu* was chosen as an action, you would be asked to type in the menu selections separated by commas in the Parameters field.

The general concept is as follows: the system always starts at row number 1, level 1. At this level a certain action is performed (sometimes needing parameters), and after this the system goes to the next level. How is the next level determined? It depends. There are two kinds of actions: those that generate a new level, and those that don't. If an action does not generate a new level, the system jumps to a level specified in the Level field, otherwise the system jumps to the level generated by the action. At this new level the same procedure is followed.

I will now discuss the three columns in the pop up window (Level, Action, Parameters):

[Level]

Two values are shown: the level, and (between brackets) the next level. If you hit <Enter> you can edit the level (except at row 1 where the level is always level 1). You are free to number the levels the way you prefer, but some structure is recommended.

If you press <F8> you can edit the next level (the level that is jumped to after the action at the current level is executed). Normally you will not edit the next level: by default it is 1, so the system will always return to the main menu (normally the action at level 1 is the main menu).

If the action specified at the current level is an action of the type that generates the next level, the manually set next level is overruled. This is indicated by <1...>. Actions generate hierarchically structured next levels. For instance, if you are at level 3 now, a *menu* action (with four menu options) will generate levels 31, 32, 33, 34. If you are at level 33 a *menu* action would generate 331, 332, 333, and 334.

[Action]

As mentioned there are some actions that generate a next level and some that don't. Actions that generate a next level are:

-Menu

next level:	(current level * 10) + X
X:	escape=0, first option=1, second option=2, third option=3,.....

-Dialog

next level:	(current level * 10) + X
X:	escape=0, first option=1, second option=2

-Sure

next level:	(current level * 10) + X
X:	escape=0, Ok=1

-TLI setlevel

next level:	X
X:	number returned as a result of a TLI expression

Some other actions are:

-Data Entry Taylor	standard input windows
-Data Entry User	user defined input windows
-TLI	to execute TLI expressions
-Quit	to quit the application
-Wait	to wait for user response
-Message	to show a message

All the remaining actions are standard Taylor menus. You can define menu access at different levels. For example: if you choose *..Detail.......* as an action, the user gets in the <Detail> menu of Taylor II, and will follow the Taylor II structure from there on. But you can also define *..Detail Elem Nr...* as an action, which is much more specific: the user gets into the editing window for a specific element immediately.

[Parameters]
Depending on which action you choose, you may have to specify some extra information in the Parameter field. For example: if you define the action *Menu* , you will have to list the menu options in the Parameter field. If you choose *..Detail Elem Nr.....* you have to specify the element number in the Parameter field. And if you define the *Quit* action, you do not have to specify anything.

3 SUMMARY

Taylor II Simulation allows for the interactive development of stand alone runtime applications. Programming skills are not required because of the unique pop up editing window with a pick list of available actions for defining the runtime application. The creation of runtimes is an effective way of getting more people to use simulation in their work. If the user does not have to build the simulation model, but only needs to input data as prompted for it, and then correctly evaluate the results, simulation can be used by people who have neither the time nor the expertise to build their own model. As always, guidance by a consultant or a key person within the organization is necessary to ensure proper simulation techniques are followed. The development of a successful runtime will prove profitable for both the creator of the runtime and the end user of the runtime.

AUTHOR BIOGRAPHY

CLIFF B. KING is the current director of customer services at F&H Simulations, Inc. He is a member of the Institute of Industrial Engineers. Formerly he was the program manager of Morton International's Japanese automotive airbag programs. He received his B.S. in Mechanical Engineering from Brigham Young University.