

CSPN GUI Requirements

The basic problem is to build a GUI for the CSPN tool.¹ Our class is to pretend it software development firm (Computer Software Solutions, Inc. [CSSI]). CSSI has just acquired a defunct software house named Borling. The major effort associated with the takeover of Borling is to make over their prime product (i.e., CSPN). CSSI has salvaged the CSPN project plan and design (and user manual) for the planned enhancements that Borling had underway. The problem is that Borling's plans and design for the GUI are much too complicated (i.e., they were planning to build five separate GUI windows). CSSI's CEO has decided that, if the project is to survive, its got to be simple to build and easy to use! A demonstration is needed for Computer World Expo in December.

The chief engineer has recommended that the original plan be scoped so as to collapse the five windows that were originally planned down to one (or two). He has suggested that the new design be malleable and extendable. In this way, it is hoped that CSSI can provide a demo that will enable them to gain important feedback from the user community. CSSI must explore how the interfaces are being used to allow for early enhancements that will better satisfy the user needs. Therefore, the chief recommends that only a small set of the total set of features that are currently available be implemented. However, the design must *account* for all of the currently available features and demonstrate how future features may easily be integrated into the interface in the future. Here are the main things that are required:

1. Deliverables

- PDR and CDR

- DNB (design method is SSA/SD)

- Demo and code

2. Development Environment (your choice) on the PC

- Java

- Visual C++

- Visual Basic

- Other

3. The GUI must be tied to the CSPN functionality.

CSPN functionality is grouped into the following partitions:

- P-CSP Language Constructs (available in the CSPN User Manual)

- CSPN Graphic command line switches (x, d)

- CSPN Verbose / Interactive switches (e, f, r, s, v)

- CSPN SPNP-setup switches (a, i, p, k, o)

¹ CSPN stands for CSP-to-Stochastic Petri nets.

CSPN Debug switches (n, t)

CSPN PN Reduction switch (c)

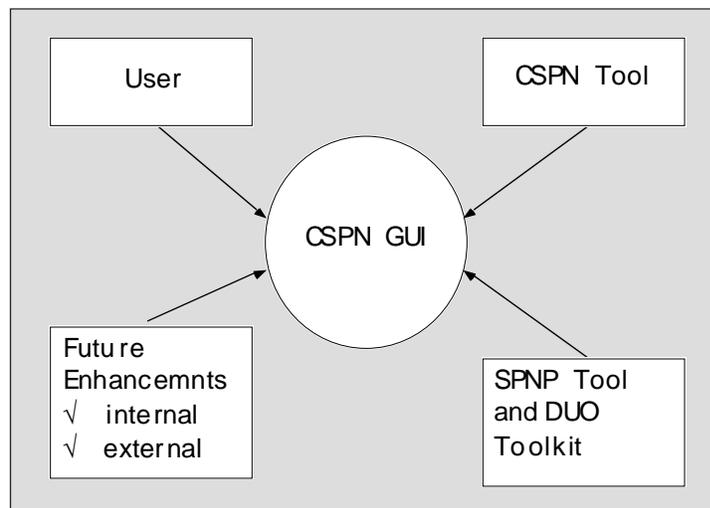
CSPN Help (h)

Running CSPN

Message window to inform user of the state of the execution (not interactive)

4. **The GUI must manage the versioning capability.**
5. **The GUI must manage the display of text and postscript (graphic) files.**
6. **The GUI must provide context sensitive help.**
7. **The GUI MUST be extensible so as to enable the incorporation of third party software tools and enhancements.**

The context diagram for the CSPN Graphical User Interface is shown here along with a picture of the vision for future enhancements (Modeling Formalisms Interoperate). This figure shows a general architecture for DOU, a modeling toolkit that promotes interoperable use of various formalisms (including at least one language(s) used in powerful model checking programs). This environment will more easily facilitate the modeling process and support the analyst with regard to prediction and assessment. The meta-language used as an intermediate canonical representation form is CSPL (C-based Stochastic Petri net Language).



Project FAQs

1. What is CSPN (CSP-to-Petri Net Translation)?

CSP (Communicating Sequential Processes) is a Process Algebra that is used to specify systems of communicating processes (parallel threads of concurrent activity) that communicate through a mechanism known as message passing. The idea is to use CSP (like a pseudo-code) to define the architecture of a system (maybe just SW, maybe both SW and HW).

Now, Petri nets constitute a different FORMAL way to describe concurrent threads of activity except that unlike CSP they are able to show dynamics (behavior) that is, they enable us to simulate the interruptions of the threads (i.e., processes) like a FSM. Or said another way, they enable us to write down all the possible ordering of events and activities that could occur given the structure of the Petri net specification (I like to call this a model). Petri nets show graphically the structure of the model (more than the CSP [which is a pseudo-code like nesting structure]) because the formal definition of Petri nets are a bipartite digraph (ask me what that means in class).

The whole point is, that CSP gives us a way to compose a system (i.e., write down like we would compose a pseudo code) into a system of communicating sequential processes, tasks or threads. The form of composition is algorithmic. We can translate (using the CSPN Tool) the functional form a stochastic form, that is, into a Petri net using standard (canonical) translation rules. Then, we can study and analyze the (stochastic) behavior of the model before we actually implement (for what we will do in class you don't need to care about how to do a stochastic analysis). We can use the model to refine our ideas about how the systems should be structured based on various external constrains (i.e., design criteria). We can predict how the system will perform and be able to make better design decisions with respect to satisfying non-functional requirements.

2. What is SPNP (Stochastic Petri Net Package)?

SPNP is a tool that takes the Petri net that CSPN creates and solves the underlying state space. The actual how does it do that question is a matter of taking the linear algebra class that is offered by the math department. I.e., how SPNP works is out of scope for our purposes. The output of the CSPN tool is the input to SPNP. The file CSPN generates is based on a language called CSPL (C-based Stochastic Petri net Language) which is a superset of the C lang. Think of SPNP as a CSPL compiler. But instead of generating an executable it generates a number like the probability of system failure is 1×10^{-9} .

3. You indicated that we should use X-windows. Is that true?

X-windows is not really accessible enough and after talking to a number of students, I've decided to let each team decide what they want to use from the following list: Java, Visual C++, or Visual Basic, or Borland. So the answer is your choice but my preference would be Java or Visual C++.

Borling's Original Needs Statement for GUI Requirements

The Graphical User Interface will consist of 5 windows using the X Window system. All five windows will be designed, implemented, tested and demonstrated using five test cases. Window 5 may be partially complete and will consist of an interface to the SPNP generated results and the version manager of window 1, the GNU plot tool. GNU plot is used to graph the differences from recordings of successive runs.

This outline is only one possibility which provides a basic partition of the necessary features that should be incorporated in the design of the windowing system. The end result window may be different but all of the windows in the system should have the same look and feel. The second figure gives an overview of the requirements for each of the five windows.

Typical CSPN GUI Window:

| | |
|---|---|
| <p>Menu and controls go here (pull downs, buttons, checkboxes, fields and help)</p> | <p>Contains the object of interest :</p> <ul style="list-style-type: none">-CSP Specification-Interactive dialog-PN Graphics-SPNP text field-SPNP results file-Plot graphics-etc. |
|---|---|

