Model-Based Code Generation: Past, Present & Future

Nicolas Rouquette, Gregory Horvath

JPL MathWorks Day
October 29 2003
Overview

- History of Model-Based Code Generation
- Past Activities
  - Deep Space 1’s 13th Technology
- Present Activities
  - Deep Impact FP System Design
  - MDS
- Future Possibilities
  - MDS and beyond...
- Conclusions
History of Model-Based Code Generation & Development
Statecharts

- Statecharts have long been used as a tool for modeling system behavior
- Using statecharts as a software modeling tool has gained popularity as of late
- Current architectural schemes promote system-level analysis
  - Component Based Architecture
  - C2SADEL – UCIs C2 ADL
- Statecharts facilitate this type of analysis
Code Generation

What is Software Code Generation?
- Write software through a mechanical process

What is Model-Based Code generation?
- Separation of concerns
  - Model: information about the application domain & architecture
  - Process: algorithms and approaches used to generate the software
Why Code Generation?

◆ Labor Constraints
  ■ Building software systems is a labor intensive process

◆ Time Constraints
  ■ Building a software system takes time, building a good one takes longer

◆ Shift in Development Focus
  ■ Concentrate on system-level design rather than implementation details
Code Generation: Past Activities
Deep Space 1 (DS1)

- Deep Space 1
  - New Millenium Project – Launched 10/98
  - Testing ground for 12 new technologies
    - Ion Propulsion
    - Remote Agent
    - AutoNav
    - MICAS
    - ...
  - Extended Mission Lasted through 2001
    - Borrelly flyby 9/01
DS1’s 13th Technology (cont)

• DS1 FP successfully employed Model-Based Code Generation techniques (13th technology)
  ■ Response code completely auto generated from statecharts with no post-processing

• Code Generation allowed for:
  ■ Reuse of the Pathfinder FP System Design
  ■ Full generation of mission-specific responses
  ■ Success despite severe time constraints
DS1 Code Generation Strategy

**Design Inputs**
- Specifications
  - Structural Definition
  - Information Flow
- Statecharts
  - Behavioral Definition

**Context Inputs**
- Software Interfaces
- Summary of all Designs

Source code, Differences, Test drivers, Cheat sheets, etc...
A Caveat...

- Code Generation seems to follow a 90/10 rule
  - 90% of user needs are met by the tool
  - 10% achievable through post-processing or custom tool modifications

⇒ Access to code generator source is essential

- DS1 FP Team performed moderate extensions to the Stateflow Coder Toolkit
  - Some of these extensions later became part of the default tool configuration
    - Lexicographic state ordering, instrumentation telemetry
Code Generation:
Present Activities
Deep Impact (DI)

- Discovery Mission
  - Fixed cost
- Scheduled for launch in January 2005
- Two Spacecraft
  - ~380kg Impactor steers itself to impact comet Tempel/1 with help of AutoNav SW
  - Flyby Spacecraft trails behind Impactor sending science data and images of impact back to earth
DI Fault Protection

- FP System uses model-based techniques developed on DS1
  - Process extended, tailored for DI
- Statecharts are still the prime modeling tool for fault responses
- DI extends the generation process to include some autonomy in the response statechart design as well
  - On DS1, all statecharts were hand-made
DI Code Generation Strategy

Three step process

1. Developer creates specification and generates 'skeleton' statechart
2. Specific behavior of response defined
3. Response code generated directly from statechart
DI Results

- For DI, code generator modification was not necessary
  - However, a fair amount of post-processing is done
- Quite efficient
  - On DI, 40 responses were defined
  - FP team = 5 members
    - Impressive for a system that interacts with every CSC
  - Same results could not be realized without code generation techniques
Mission Data Systems (MDS)

- Attempt to develop a widely reusable core set of FSW components
- Includes flight, ground, test, and sim capabilities
- CAR based system design
  - CAR = Component ARchitecture
- Models are key!
MDS Code Generation Concepts

- MDS uses code generation system-wide, not just in a particular subsystem
  - In fact, MDS has no notion of subsystems
- Flexibility of Stateflow allows for a myriad of applications
  - Stateflow defines a C-language target
  - MDS defines custom targets to arbitrary languages
    - MDS-style C++
    - Java
    - XML
FIGURE 1: Sample view of XML Exporter in the Design and Implementation Process
Code Generation: Future Pursuits
XML Code Generator

- Extension to Stateflow Coder Toolbox
- Creates tool-neutral representation of source statechart
  - Includes information about Stateflow-defined order of evaluation of
    - Multiple parallel child states
    - Multiple transitions emanating from a common source node
  - Partitioned such that analysis can exclude Stateflow-imposed properties if desired
XML Code Generator (2)

- XML descriptions are often not the final product
  - Can theoretically use XML descriptions to generate code in any language

- XML artifacts permit real-time decoding of MDS instrumentation telemetry
  - All necessary preprocessing done during XML generation – No extra steps necessary
MDS Themes

- In MDS, Models are central
  - Models tie everything together
- MDS will use models as the basis for configuring a FSW deployment
- Reuse of a complex system like MDS necessitates automation
  - Code generation helps keep down development costs
Conclusions

- Model-based design and system-level analysis becoming more prevalent
- As software systems increase in size, automation becomes essential
- Marriage of the two concepts presents a powerful method for system design, implementation, and analysis required by the high-performance, complex systems of tomorrow
Links

Deep Space 1 Mission Homepage
- http://nmp.jpl.nasa.gov/ds1

Deep Impact Mission Homepage

MDS Homepage