Summary of Tools Breakout Sessions

1. Use of tools implementing formal methods should be started very early, even before the requirements are formalized. The tools should then be used consistently throughout the software life cycle.

   Rationale: Using formal methods to model the requirements can expose inconsistencies and missing requirements. Results from applying formal methods can be used in communicating with customers. A real life example of this was described A caution in speaking with customers is that the discussion should be in terms that the customer understands.

   Using formal methods after the software has been developed is not nearly as effective as starting with formal methods at the architecture level. In fact, the architecture needs to be compatible with formal methods. We need tools that provide a principled look at how to build systems. To bridge between design and code, the formal methods tools should be used to generate test cases. Tools also need to be used to demonstrate that configuration data is correct.

2. We need to make formal methods tools more accessible. Tools should span model types and levels of formality. Tools at different levels of formality, as well as tools used at different stages of the life cycle (i.e. tool chains) should be integrated and interoperate wherever possible.

   Rationale: Results from static analysis tools can assist in the formal methods proofs. To make tools more accessible, it was suggested that formal tools plug into an environment such as Eclipse. An example of a tool that people are more familiar with is Simulink. An additional concern for hybrid physical/computer systems is addressing domain-specific concerns, including physics and biological constraints.

3. To help users with the methodology of using formal methods, a forward and backward chaining knowledge based framework was suggested for expressing the methodology.