	ASL/NSL	MissionLab	Integration Issues
Modeling	Schemas are the basic unit of modeling (Percpetual, Motor, and Coordination Schemas). Assemblages are recursive schemas. Neural Networks serve as implementation or structural schemas (Neural Schemas). Further detailed neural levels are in development.	Schemas are the basic unit of modeling (Percpetual, Motor, and Coordination Schemas) for continuous coordination tasks. FSA assemblages provide recursive schema specification, and can be defined also in terms of continuous coordination tasks.	There is an overlap at the continuous coordination modeling level. In contrast, ASL lacks FSA modeling, while MissionLab lacks neural networks modeling.
Simulation Language	ASL defines the model schema topology, calling predefined schemas. NSL helps define new schemas (either in C++ or Java).	CDL defines the model at one level, and later generates CNL code for the AuRA architecture or MRPL for the ARPA UGV architecture.	Both systems distinguish between architecture dependent and independent languages.
Architecture Independent	ASL describes the schema hierarchy from predefined schema definitions; can link to heterogeneous schema implementations. A Tcl-based script language provides limited ASL functionality.	CDL describes the schema hierarchy from predefined schema definitions. Compilation generates architecture specific CNL or MRPL code. A script language, CMDL, provides limited CDL functionality for MissionLab.	Both systems provide an independent architecture language, although CDL compiles into a second language, as opposed to ASL. Both systems provide script languages for interactive commands.
Architecture Dependent	New schema definitions are written with the help of NSL, either neural networks (neural schemas) or general purpose C++ or Java schemas. NSL schemas can then be linked into the system library to be later called from ASL.	CNL defines new procedures, as an extension to C++, as well as schema topologies. A user may write direct CNL code, or generate it from a CDL specification (only predefined schemas at the CDL level).	Both NSL and CNL are C++-based. The main difference is that NSL defines only new schemas, while ASL defines the schema topology. CNL does both.
Implementation Language	NSL neural schemas and general purpose code is implemented in C++, while a Java implementation is currently in development.	CNL code is implemented in C++.	Both share C++ as implementation language.
Execution Environment	ASL/NSL run only simulated, primarily in Unix, although ports are being made to Win95 and MacOS. The Java version enables Web execution.	MissionLab can run simulated and also in a real-time autonomous robot environment.	Both share a simulated environment.
Graphics	Latest graphics development is in Tk (previously in Motif, and XView before that). The Java implementation has its own graphics.	Graphics are in Motif.	They have different graphics capabilities, sharing Motif in Unix domain. NSL Tk graphics port helps multi-platform migration (PC and Mac)
Synchronization and Communication	Light weight processing with p-threads. Interprocess communication via datagram sockets (future PVM or MPI implementation)	Light weight processing with C-threads. Interprocess communication via IPT.	Standardization needed, or design platform independent interprocess communication.