ISG: Instantaneous System and Software Generation

Main Features

- Multi-team and multi-site capability
- Support of domain of distributed and/or real-time systems, embedded systems
- Full domain coverage: all functional and non-functional requirements by only one specification and design language
- Support of synchronous and asynchronous systems
- Generation of instantaneously executable infrastructure for distributed / real-time systems from a specification in some 1 .. 10 minutes
- Automated integration of external Ada or C source code
- Auto-generation of missing functions for immediate integration
- Auto-stimulation on behavioural and functional level
- Incremental refinement: ”specify and get immediate feedback”
- Support of heterogeneous OS (different OS on distributed nodes)
- Tailoring to customer-requested operating systems possible
- Immediate visualisation of system properties on development and/or target platform
- Support of Model-Based Testing (MBT), auto-generation of tests
- Optionally, automated verification of the code generator for each application

Benefits

- Very short turn-around time from specification to execution on the target system
- Agile, but automated and support of immediate verification and validation
- Higher flexibility in changing and maintaining system properties
- Shorter time-to-market
- Easy and inexpensive evaluation of system architecture
- Verified specification (”no code from a faulty specification“)
- Outmost support of validation by visualization of system properties of a system after execution on the intended platform
- Filtering of information
- Graphical presentation of system properties
- Numerous metrics for assessment of quality

BSSE System and Software Engineering

ISG: The fully automated process chain
from the specification to execution on the target system in minutes

verified models and code – validation by visualized recorded properties
higher productivity and efficiency – shorter time-to-market
drastically reduced costs – higher quality – less risks

Quantitative and fully independent assessment of a specification
Contractual problems through imprecise requirements disappear
Clear structures eliminate long/complicated discussions
Visualisation of all contents and interrelationships
Immediate hints on required corrective actions
Automatic reporting (Multi View)

Version delivered in 2000 generated the infrastructure for the Material Science Laboratory (MSL) successfully operating on-board of ISS

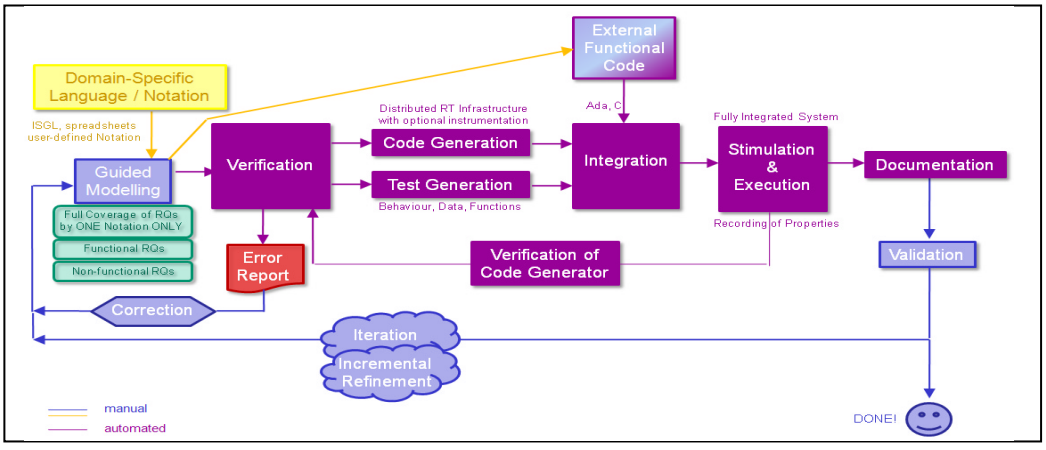
Instantaneous System and Software Generation (ISG) fully automates all steps from delivery of a specification to system execution on the target system for a distributed and/or real-time system.

The specification is expressed in a system-level notation, as close as possible to the user’s world. Default notations are ISGL (ISG Language) or even spreadsheets. This allows a non-software expert to build complex and large software systems, while always knowing from immediate feedback what was really specified. The short turn-around times and the capability for incremental refinement and iteration support an agile approach equivalent to continuous maintenance right from the beginning.

An important aspect is the verification of the specification. It is checked against a set of rules which prevent generation of erroneous code from an erroneous specification, so that a user can be sure that the code always corresponds to the specification. Due to automatic stimulation the properties of the system are recorded, filtered and visualized, allowing an immediate assessment on whether the specification really expresses what a user had in mind.

The Process

- **Domain-specific support:** the limitation of a process to a certain domain allows maximisation of the degree of automation while still supporting an infinite number of applications of that domain. In consequence, no manual intervention is required for the build-process after delivery of the model and additional external source code.
- **Guided modeling:** the domain-specific notation guides a user the right way. An underlying meta-model enforces use of the relevant specification elements in proper context.
- **Verification:** the model is checked against a set of domain-specific rules ensuring completeness, consistency and correctness.
- **Code generation:** The model is transformed into the specified environment including the interfaces to the (real-time) operating system.
- **Test generation:** Test inputs – both nominal and non-nominal – for stimulation of the system and its functions are automatically derived from the model.
- **Stimulation & execution:** The (distributed) system is automatically installed on the target system and its properties are recorded.
- **Documentation:** The recorded information is processed and filtered, the system properties are presented as text, tables and graphics, both in compressed and detailed form.
- **Verification of the code generator:** The observed properties are automatically verified against the input model.
- **Validation:** the user only has to assess the observed properties and to conclude on whether this is what is wanted.



ISG: Instantaneous System and Software Generation

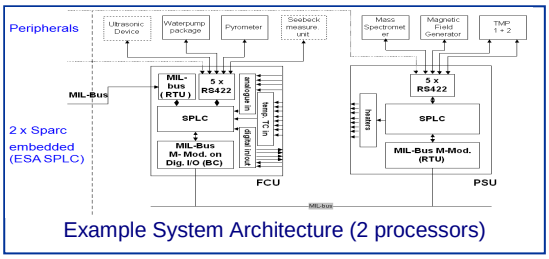
Modeling & Code Generation

The modeling language allows defining the behavioural, data-flow and non-functional requirements of a distributed and/or real-time system. From such a specification code is generated which provides everything to run the application on a distributed / real-time platform incl. all interfaces to an (mini) operating system.

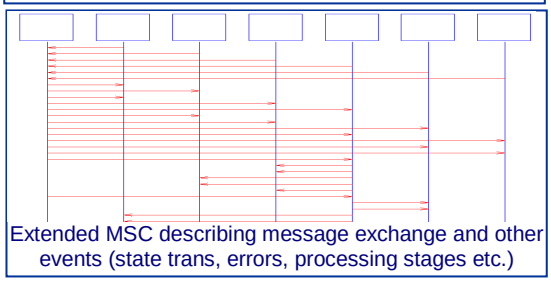
Specific application functions are to be delivered as external source code (Ada, C) which will automatically be integrated.

System Complexity

The methodology is fully scalable in size and complexity. Proven up to 16 processors, 40 processes, 1000 message types, 400 states, 1500 state transitions, 10000 atomic actions in state transitions.



Example System Architecture (2 processors)



Extended MSC describing message exchange and other events (state trans, errors, processing stages etc.)

Real-Time Processing

Asynchronous processing of data and events, generation of sporadic or cyclic events, creation of timeouts, monitoring of deadlines, timeouts and resource consumption.

Distribution

Automated and transparent distribution, easily configurable by high level directives. The generated system can be executed on any number of physical processors between one and the maximum number of allocated processors without any need to change inputs other than definition of the nodes. Automated set-up of the processor network and harmonisation of time between processors.

Verification & Validation

checks at pre-run-time, run-time and post-run-time, model checking on FSMs, checking of logical and performance properties against model specification.

Process Interfaces

The ISG process may interface with

- **Tools and methods:** bridges are supported for UML and AADL (others on request)
- **External software** by its integration on source code level (Ada, C)

Fault Injection

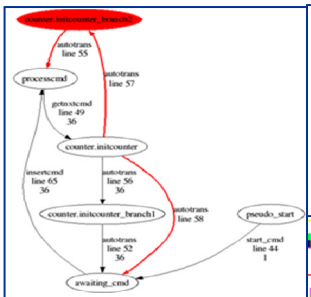
Loss of data, generation of illegal messages, stimulation of dormant states
time jitter on timed events (one-shots or periodic events, various time jitter modes)

Fault Tolerance

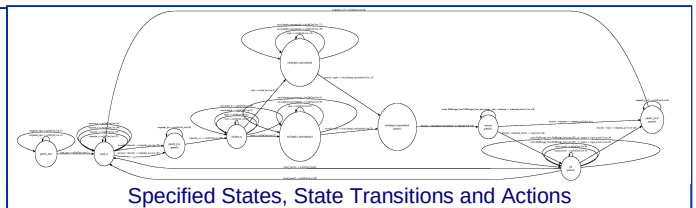
Support of redundant processors and communication channels, automatic channel switching, no “duplicate message” problem

Reporting & Visualisation

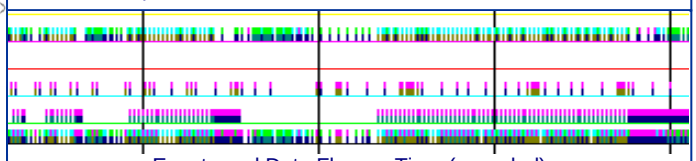
reports on logical and performance properties, exceptions and injected faults according to the instrumentation selected by a user, presentation of states and state transition, data flow and events in a MSC-compatible format and versus time, more versatile tables graphics in detail and compressed shape. Reports extensible to user needs.



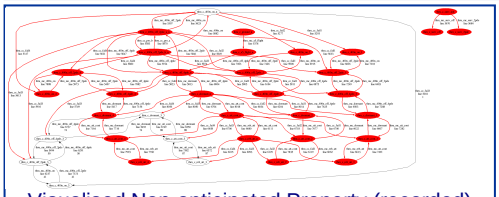
Visualised Non-anticipated Fault (in red colour, recorded) non-covered states + trans. Indicate a principal design fault



Specified States, State Transitions and Actions



Events and Data Flow vs.Time (recorded)



Visualised Non-anticipated Property (recorded) non-reachable sub-nets in a Finite State Machine

Multi-team and multi-site capability

In ISGL a model can be divided into parts which may be processed

- **by different teams / engineers:** just before auto-generation the parts have to be merged. Interfaces and dependencies will be checked automatically and be marked as erroneous – if so.
- **at different sites:** to be merged as above; the user’s existing infrastructure can be used for shared access.

Integration into a Customer’s Infrastructure

As far as possible without compromising the ISG inherent guarantee on correctly generated code, the ISG process can be integrated with customized interfaces not yet supported on request.

Message / Data Exchange

Any messages as defined by the user, support of any user-defined data formats and data, an incoming message is treated as an event. Automated conversion Big-Little Endian for any format.

Status and Mode Control

is based on Finite State Machines, mandatory exception handling

Maintenance Support

Capability for easy and inexpensive re-structuring without losing user-provided code, support of incremental development, prototypes can be transformed into final product without having to start over

Stimulation

Messages and events, automated generation of CPU load and data traffic

ISG

- runs on Un*x platforms (Solaris, Linux)
- generates C code, integrates automatically C and Ada source code
- supported target platforms: Un*x and VxWorks; PC, Sparc
- support of other development and target platforms on request
- “bare machine” on target possible

Documentation

Reports are automatically generated in RTF format. Errors are reported in text and graphics

Further methods and tools

- Systematic Requirments Management (SRM)
- Systematic Project Planning (SPP)
- Fully automated testing (Ada, C) DARTT and DCRTT

Support services from BSSE:

- **Support** in establishing models
- **Transfer** of conventional text specifications to models
- **Re-engineering** of legacy systems with ISG
- **Quality analysis** and quantitative assessment of specifications that already exist
- **Training** to apply ISG
- **Complexity and feasibility analyses** from independent, neutral reviewers based on an ISG model
- **Early risk identification** through using ISG

BSSE System and Software Engineering

Dr. Rainer Gerlich - BSSE System and Software Engineering

Phone : +49 (0)7545 911258
Fax : +49 (0)7545 911240
E-mail : info@bsse.biz
Web : www.bsse.biz