

# Automated Software Production

## An Overview

Status: June 2002

Dr. Rainer Gerlich  
Auf dem Ruhbühl 181  
88090 Immenstaad  
Germany

Tel. +49/7545/91.12.58  
Fax +49/7545/91.12.40  
Mobil +49/171/80.20.659  
e-mail [gerlich@t-online.de](mailto:gerlich@t-online.de)

---

## Contents

- Overview on ASaP  
Automated Software Production
- Space-Specific Support, Real-Time and Distribution
- Effort Reduction during Development and Maintenance
- Verification & Validation, Quality Issues
- Resource Monitoring and Risk Reduction
- How to use
- Products and Services

# Overview on ASaP

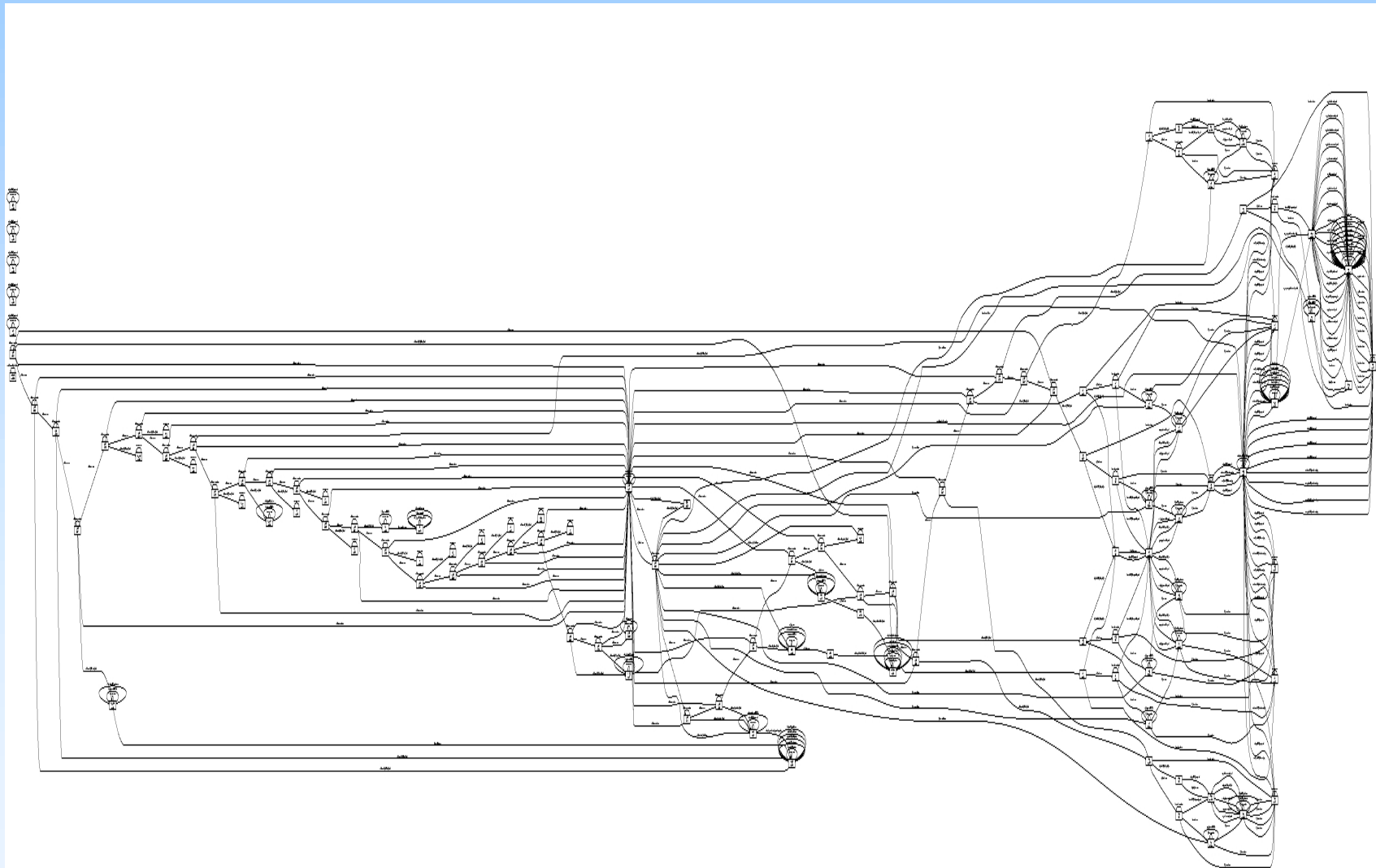
## The Goals of Automated Software Production

- to produce systems on system engineering level
- to reduce risks
- to save costs during development and maintenance
- to shorten development and maintenance time
- to ensure correctness

## Production on System Engineering Level

- system production without education in software engineering
  - system definition** by literals and figures only
    - communication by messages
    - operational states
    - performance constraints (timeout, period, time jitter)
    - resource constraints (buffer size)
    - fault tolerance (channel properties)
    - topology (nodes)
    - V&V options (fault injection, stress testing)
    - properties of data (HK, telemetry, ancillary data)
- automated installation and execution
- automated evaluation of system properties
- automated documentation of system definition

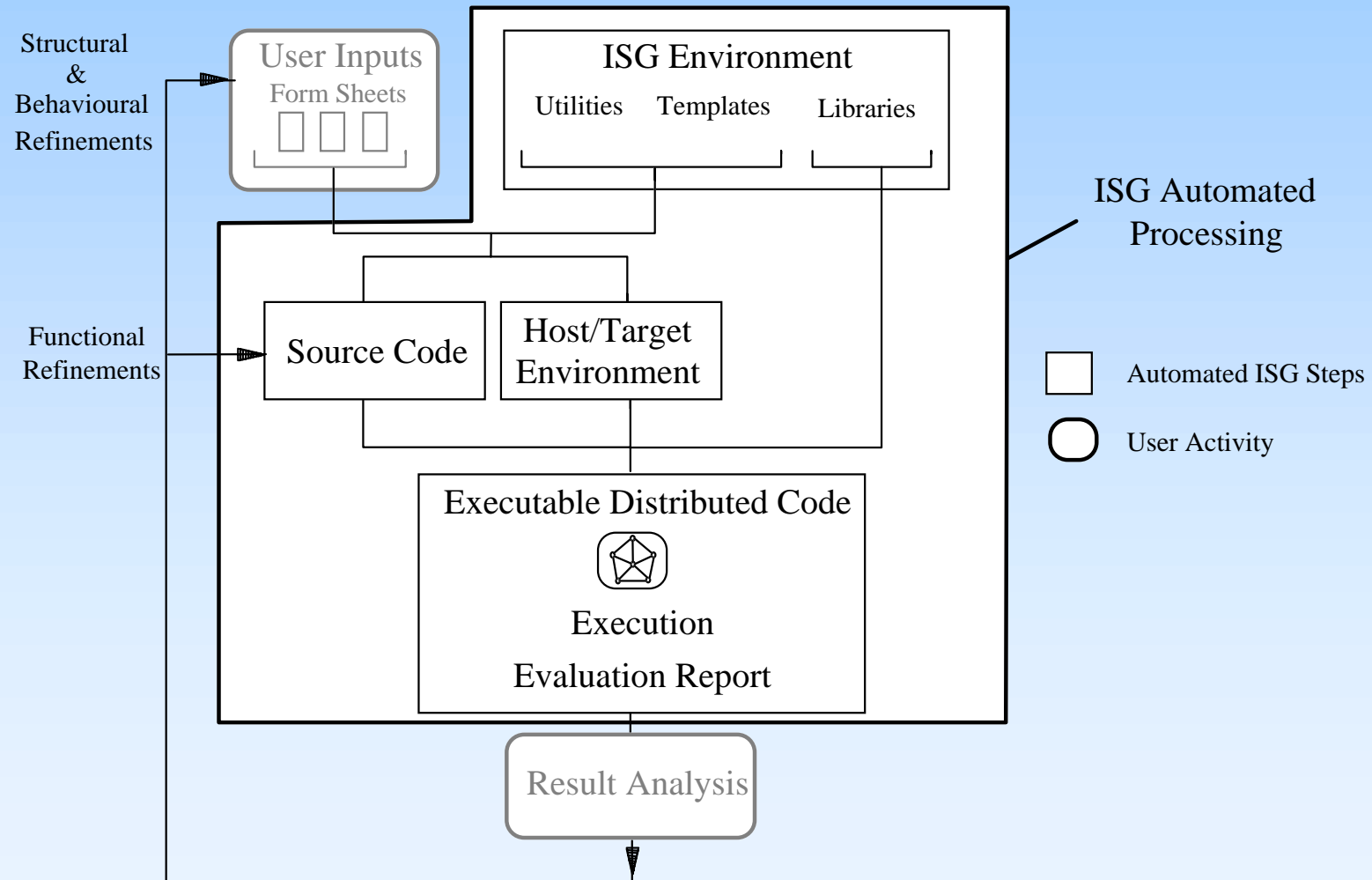
# Complete Data Flow Of A Distributed Real-Time System (Type DMS)



## What is Covered by ASaP

- infrastructure of a distributed / real-time system
  - ISG
  - Instantaneous System and Software Generation
  - complemented by sequential software (algorithms)
- sequential software
  - reusable automated production chains
    - ☆ command handling, data acquisition, telemetry handling
    - database software
  - generic environment for „operations on user-defined types“
  - existing software
  - software generated by other tools  
MathLab, Scade

# Automated Production Chain of ISG



## Risk Reduction

- immediate feedback on system definition
  - feedback from executable system within some ten minutes
  - feedback from real target
- on acceptance of system definition:  
guarantee to get a correct system
- support for
  - test instrumentation
  - fault injection
  - stress testing
- identification of operational bugs
- identification of resource consumption
- support of topology and platform subsets

## Cost Saving

- Limitation of effort
  - dependency on ***size of system definition only***  
not on size of implementation
  - dependency: constant effort or first order
- changes on level of literals and figures only
  - processes, messages, states, performance figures
  - topology (allocation of processes to processors)
  - platform (allocation of processors to OS and processor type)
- fully automated production chain
  - no human intervention required for intermediate steps

## Equivalent of Man Power and Costs

Application	Generation Time	LOC	Equivalent Man Power	Equiv. Man Years / 1 h Generation Time	Equiv. Costs / 1 h Generation Time
Distributed Real-Time System	20 minutes	40,000	2.5 man years	<b>7.5</b>	<b>600 kEuro</b>
Data Processing and Database Software	10 minutes	5,000	500 man hours	<b>2</b>	<b>160 kEuro</b>
Distributed Synchronous System	10 minutes	20,000	1.25 man years	<b>7.5</b>	<b>600 kEuro</b>
Operations on User-defined Types	1 second	200	20 man hours	<b>&gt;36</b>	<b>&gt;3 Mio.Euro</b>
Interface adaption + On-Line Help Facility	30 seconds	15,000	1 man year	<b>120</b>	<b>10 Mio. Euro</b>

## Shorter Development Time

- Executable system immediately available within some 10 minutes after delivery of inputs
- automated production chain provides correct system
- automated testing inherently covered
- automated support of platform migration
- automated evaluation of system properties for system acceptance

## Inherent Correctness

- proven production rules ensure correct system construction
- known fault rate due to absence of manual intervention
- high degree of reuse of construction rules, templates and library
- prevention of fault propagation by assertions
- check of user inputs for completeness, correctness, proper exception handling

**Space-Specific Support**  
**Support of Real-Time Processing and Distribution**

## Handling of Ground Commands

- Conversion of high-level ground commands into time-tagged on-board sequences
  - absolute or relative start time
  - correlation to on-board commands just by
    - ☆ associating the HLC literal with an on-board command literal
    - ☆ assigning a sequence number
    - ☆ specifying a timeout for each sub-command
  - check of conditions required for command execution
  - automated generation of and/or link to checking functions
- Command Handler
  - automated generation of command handler
  - customisation of command handler
- Interface to up- and downlink

## From Data Acquisition To Telemetry

- Support of combined definition of hardware and software properties of data
  - e.g. by a spreadsheet already used for hardware description
  - automated generation of acquisition scheme and data addresses in memory
  - automated generation of calibration and limit monitoring functions
  - automated generation of firmware table
- Tuning of data acquisition
  - bursts of data if not scattered
- Telemetry
  - automated generation of telemetry frames from spreadsheet inputs
  - automated sending of frames

## On-Board Database

- automated construction of distributed database from spreadsheet information(e.g. Excel, Access)
  - data types for data acquisition and telemetry
  - data arrays according to acquisition frequency
  - data description and associated processing functions
  - allocation of memory (RAM, ROM, NV-RAM etc.)
- database update
  - automated update / data exchange between distributed parts
- emulation of distributed database on node sub-set

## Real-Time Processing

- automated set-up of real-time activities from data information
  - timeout condition with exception handling
  - asynchronous events
    - ☆ start after tbd time/cycles and execute tbd cycles
  - periodic activities
    - ☆ start after tbd time/cycles
  - stop cyclic activities, reset timeout
- evaluation of robustness and handling of overloads
  - generation of time jitter
  - scaling of time
- feedback
  - execution time, response time, cycle overruns, timeouts

## Distribution

- automated set-up of distribution from data information
  - identification of node properties (OS, version, processor type)
  - generation of specific executable for each node case-by-case
  - distribution of processes and control of execution
  - communication ports and protocol
  - conversion of binary data
  - harmonisation of time
  - monitoring of nodes
- transparent distribution
  - change of distribution does not effect data and control flow definition
  - sub-set of topology possible up to one node only without change of user inputs
  - arbitrary allocation of processes to nodes even for heterogeneous platforms



## Automated Graphical Documentation

- distribution and platform properties
- control and data flow
  - what happens when a message arrives
  - which process(es) does send the message, where does it come from
  - complete data and control flow
- behaviour
  - states and state transitions
- function hierarchy
  - caller-called
  - called- caller
  - complete calling hierarchy
- automated merge of user text

---

## **Effort Reduction during Development and Maintenance**

---

## Development and Maintenance

- support of incremental development and refinement
- development = continuous maintenance

## ScaPable Software

### ■ Scalability

- an infinite set of system configurations / topologies is supported for a certain application domain like  
embedded systems, real-time systems, client-server systems
- no manual implementation effort is required to correctly generate a certain configuration from
  - ✧ provided literals, figures and
  - ✧ templates and data types

### ■ Portability

- a certain set of platforms (processor type, operating system) is supported
- no manual effort is required to map the system onto such platforms or to move from one platform configuration to another one

## Limitation of Verification and Validation Effort

- Verification effort does not depend on the size of the implementation
  - inherent correctness by construction rules: no human effort
  - automated verification of inputs: no human effort
  - limited number of items to be verified
    - very limited set of basic C types, not an infinite set
    - „being correct once, being correct for ever“
- Validation effort is reduced by automated evaluation of system or software properties
  - automated test data generation, fault injecton, stress testing
  - automated checks on system properties
- Human involvement depends on zero or first order of the **specification size**, and not at all on implementation size

## Changes (1/2)

- process structure
  - copy-cut-paste of formsheet lines
  - grouping of actions
- control and data flow
  - literals of processes, messages, states, processing functions
  - list of receivers (process literal, instance)
- timing
  - period and timeout intervals

## Changes (2/2)

- topology
  - associate process literal and instance with logical node literal
  - associate logical node literal with (IP-)address of physical node
  - automated identification of actual sub-set of nodes including „one-target“ platform
- channels
  - wild-card definitions or dedicated definitions
  - association with physical channels (bus, UDP, IPC, file, screen)
  - type of fault tolerance (grouping of redundant channels on literal level)
- platform
  - specify cross-development platforms

## Integration

- integration = change of topology or platform
- change of topology
  - target system not available
    - ☆ use EGSE or workstation
  - target system not fully available
    - ☆ map logical nodes on available physical nodes  
possibly scale time to cover the increased workload
    - ☆ add EGSE or workstation(s) to replace missing targets
- change of platform
  - specify new cross-development platform  
which logical nodes reside on which physical node
  - specify new mapping between logical and physical nodes

---

## Verification & Validation, Quality

## Verification

- user inputs (literals, figures)  
completeness, correctness, platform availability
- automated production steps  
was a result generated?
- inherent assertions
- fault injection  
illegal messages, loss of messages
- automated instrumentation for reporting
- automated generation of textual report
  - coverage
  - non-executed messages and states
  - exceptions
- flexible / easy platform migration

## Validation

- automated instrumentation for reporting
- automated generation of textual and graphical reports
  - resource and performance figures and profiles
  - message sequence charts with debugging/internal information
  - timing diagram with data flow
- fault injection
  - time jitter
  - overload by scaling of time

## Quality - Measures

- correctness by construction rules  
„being correct once, being correct for ever“
- correctness by reuse
  - automated production = 100% reuse
  - bug in a template → multiple occurrence → good chance for fixing
- correctness by assertions and checks
  - prevention of fault propagation
  - exception handling requested by ISG
- coding and compilation standards

## Quality - Results

- Two bugs reported by the user (applications #1 and #2)
  - since delivery of first ASaP / ISG system 18 months ago including EM integration (= final integration of software)
  - bug #1** overflow of (wrap-around) command counter after injection of  $2^{15}$  ground commands at a rate of about 1 command / sec (about 8 hours of continuous system execution)
  - bug #2** wrong assignment of process priorities in case of distribution
    - in total 45,000 LOC
- resulting initial bug rate  $< 10^{-4}$  / LOC
  - but please keep in mind: these bugs do not depend on #LOC
  - we also could have generated 2,000,000 LOC  $\rightarrow 10^{-6}$  / LOC
- literature:  $10^{-3}$  / LOC: very good

---

# Resource Monitoring and Risk Reduction

## SPLC / VxWorks Exercise (1/2)

- first project on SPLC / VxWorks platform
  - memory: 6 MB
  - Sparc 14 MHz
  - estimated sizing budget, no timing budget
  - considered as critical
- first results on work station
  - may fit with sizing and timing constraints,  
but **not** far beyond critical limits

## SPLC / VxWorks Exercise (2/2)

- first tests on single processor
  - sequential software: stubs
  - sizing budget: no problem
  - timing budget: still critical
    - but can be solved by additional effort or more ground processing
- first tests on real target / two processors
  - sizing budget: still no problem
  - timing budget: still critical
- first tests with database software
  - out-of-range of timing budget
  - solution: see data processing exercise

## Data Processing Exercise

- Database contents
  - about 600 data items
  - calibration, limit monitoring, processing, telemetry frame generation
  - estimation:  
could impose heavy load
- first tests with full database / data processing
  - sizing budget: still no problem
  - timing budget: very critical
- solution
  - re-order calibration and telemetry groups
  - contents of ONE column only changed in spreadsheet
  - automated re-generation of database software
  - problem of timing budget solved without need for additional tests

## Verification on Node Subset

- node subset
  - only one of two target processors available
- urgent change
  - change of database structure
- test and verification
  - test on node subset / one processor
  - emulation of two-processor system
  - updates of distributed database on one processor
- migration to full target
  - two target processors available again
  - automated generation for new configuration
  - running correctly on full target immediately

---

# How to Use

## Provision of a Data Management System

- provided by Scapable Software (ASaP/ISG)
  - handling of ground commands and transformation into (time-tagged) on-board command sequences
  - handling of the complete chain from data acquisition to telemetry frame generation
  - distributed database
  - real-time scheduling
  - inter- and intra-process communication
  - creation of the distributed executables, distribution, execution
  - test case generation, fault injection, stress testing
  - instrumentation and result evaluation (coverage, performance, ...)
- remaining
  - specific algorithms and firmware
  - may also be covered by ASaP like the distributed database

# Provision of an AOCS / GNC System

Attitude and Orbit Control, Guidance and Navigation

- provided by Scapable Software (ASaP/ISG)
  - as for DMS
    - handling of ground commands and transformation into (time-tagged) on-board command sequences
    - handling of the complete chain from data acquisition to telemetry frame generation
    - distributed database
    - real-time scheduling
    - inter- and intra-process communication
    - creation of the distributed executables, distribution, execution
    - test case generation, fault injection, stress testing
    - instrumentation and result evaluation (coverage, performance, ...)
- remaining
  - specific algorithms
    - (automatically) plug-in output from tools like MathLab or Scade
  - other algorithms may also be covered by ASaP

## Definition of Control and Data Flow (1/2)

- principal elements
  - processes
  - exchanged messages
    - ☆ incoming, outgoing
  - states (FSM)
    - ☆ one state may be sufficient
    - ☆ initial state, final state
- I-P-O  
**Input - Processing - Output**
  - input: incoming message associated with initial state
  - processing: user-defined function (sequential code, ASaP)
  - output: outgoing message associated with a state
    - ☆ same process: state transition or same state
    - ☆ receiving process: state in which message is expected
- sequences of I-P-O lines, groups per incoming message

## Definition of Control and Data Flow (2/2)

- Instances
  - # instances of a process
- I-P-O activity of a process type
  - input: data triple
    - ✧ process (literal)
    - ✧ initial state(literal)
    - ✧ incoming message (literal)
  - processing function:
    - ✧ function name (literal)
    - ✧ if not existing a stub will automatically be generated
  - output: data quadruple
    - ✧ destination process
    - ✧ list of destination instances (\*, n, 0, RTS, specific list)
    - ✧ outgoing message
    - ✧ expected state / final state

## Real-Time and Performance

- timeout
  - range (min, mean, max)
- cyclic activity / period
  - range (min, mean, max)
  - type of time jitter
  - start event
  - # of cycles or periodic
- amount of data
  - as far as stubs are plugged-in
  - range (min, mean, max)
- estimated # of logical processor cycles
  - range (min, mean, max)

## Topology and Distribution

- communication channel
  - logical channel to be used for outgoing message (literal)
  
- allocation of instances to logical nodes
  - mapping list or single node (literal)
  
- distribution
  - mapping of logical nodes to physical nodes  
literal vs. IP-address or alias
  - cross-development information if any  
process literal and target id

---

## Products and Services

## Products and Services (1/3)

- ISG  
Instantaneous System and Software Generation
  - current platforms
    - ☆ Solaris, Linux, VxWorks
    - ☆ DSP OS on request
    - ☆ Sparc, SPLC, PC
  - tool and training
  - consulting
  - tool and project start-up (fixed price)
  - work-package / deliverable (fixed price)

## Products and Services (2/3)

### ■ ASaP

#### Automated Software Production

production of sequential code by generic or customised packages

- platforms
  - ☆ C
  - ☆ other languages on request
- packages for operations on user-defined types
- from data acquisition to telemetry handling
- command handling
- distributed memory-resident database

### ■ Consulting

organisation of automated software production

## Products and Services (3/3)

- Integration of ASaP / ISG with existing environment
  - important:
    - users of ASaP / ISG can keep an already existing environment
  - an interface between such an environment and ASaP / ISG can be provided