BSSE System and Software Engineering

Benchmarks on Automated System and Software Generation

Higher Flexibility, Increased Productivity and Shorter Time-To-Market by ScaPable Software

DASIA 2002
- Data Systems in Aerospace -
Dublin, Ireland
May 13 - 16, 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

Facing the Current Situation of SW Development

- A handcrafted product of daily life
  - expensive, but we are appreciating an individual shape
  - the product is unique by „bugs“
  - we are proudly taking such bugs as an indicator for high costs and uniqueness
- A handcrafted software product
  - also expensive, unique by bugs, buggy by handcrafting
  - same attributes, BUT ...
  - we would like to get it cheap and without bugs
- solution in daily life
  - keep „man“ out of the loop
  - take benefit of automated production chains which can continuously be improved

Benchmarking - A Pre-Condition for Improvement

- A variety of methods and tools are on the market
- But what about benchmarks ?
  - only rarely available
  - when available, are they derived from a real project?
  - how representative they are really?
- What about productivity ?
  - productivity is out-of-scope
  - current costs are accepted as a matter-of-fact
- What about quality ?
  - it is known that „man“ introduces the bugs
  - hence „man“ is the biggest problem
    because we do not have sensors for evaluation of software properties
  - but it is believed: more man-power can solve the problem
- Situation for methods and tools is like for some medicine
  - **it is believed they will help, although a real proof is missing**

Current Status and Potential Improvements

- Emphasis is put on feedback by documentation
- Assumptions (believed, but proof is missing)
  - better readability of documentation guarantees success
  - just playing with the system guarantees success
    (like use cases, prototyping on non-representative platforms)
- most problems are related to later implementation
  - platform, HW and SW environment, integration, performance
  - such problems cannot be covered at all by above policy
- potential solution like in daily life
  - introduce automation (cheaper, faster, earlier reductionof risks)
  - keep „man“ off from the critical tasks like implementation
  - limit the influence of „man“ to system engineering

Automated System and Software Generation

- The **FULLY** automated approach
  - only describe a system by literals and figures
  - only provide templates and data types
  - only press a button and

    generate a system or software by construction rules
  - always generate a correct system giving a real feedback
  - human interaction
    - only at beginning and end of production chain
    - this makes generation process highly efficient
- Generation process similar to spreadsheets
  - input fields from which results are derived by construction rules
  - immediate feedback on results whenever input changes

Featuring SPQR

- SPQR
  - it is not
    Senatus PopulusQue Romanum
- BUT
  - **S**calability
  - **P**ortability
  - **Q**uality
  - **R**isk reduction
- Scalability and Portability

$\rightarrow$ **ScaPable Software**

ScaPable Software

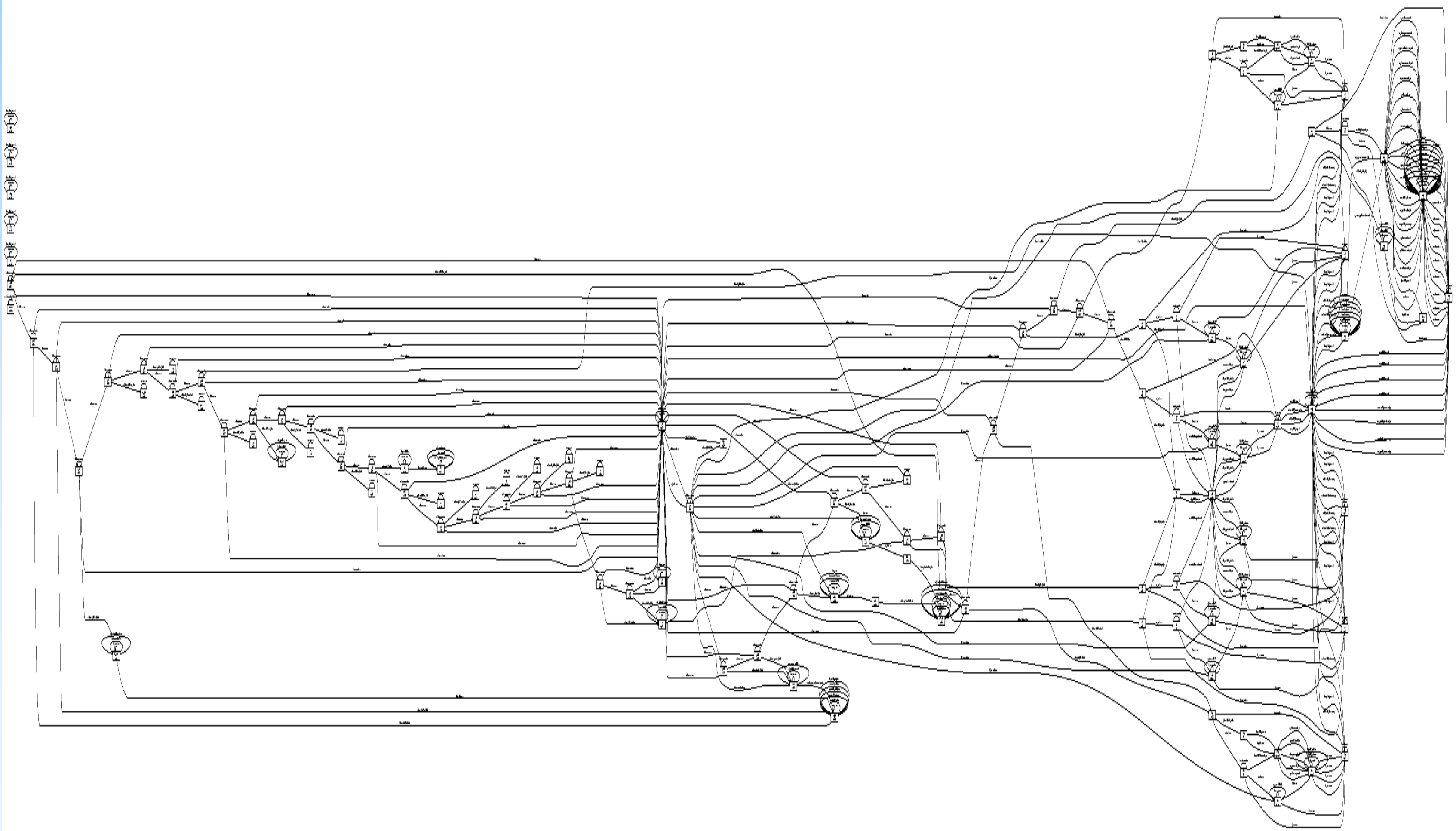
- Scalability
  - an infinite set of system configurations / topologies is supported for a certain application domain like
    embbeded systems, real-time systems, client-server systems
  - no manual implementation effort is required
    to correctly generate a certain configuration from
    - provided 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 application
  - 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

BSSE System and Software Engineering

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



More Examples (1/2)

- Critical Control System (type: AOCS / GNC)
  - environment for stress testing and fault injection
  - distributed system:
  - 16 processes mapped onto 1 .. 16 processors
  - data buffering (toggle buffer), synchronous processing
  - get each possible configuration within about 10 minutes
- on-board data processing
  - from data acquisition to telemetry frame generation
  - distributed on-board database with continuous updating
  - get it within about 10 minutes for about 600 data items

More Examples (2/2)

- operations on user-defined data types
  - provide the types and the templates
  - get all the derived classes within seconds
  - e.g. random initialisation, printing, data conversion
- adaptation of a language interface, automated documentation, test and training environment
  - provide the types and the function prototypes for 500+ functions
  - get all glueing software
  - get all types and prototypes for the other language
  - get interface documentation for the User‘s Manual
  - get a test environment for the 500+ functions
  - get a help facility which provides context-dependent function parameters
  - get everything in less than 2 minutes

BSSE System and Software Engineering

	Application	Size of Input lines / bytes	Size of Output lines / bytes	Durat.
DMS	**Distributed real-time system** 2 processors, 40 process types	1600 150,000	484,000 20.0 MB	20 min
	Data processing and database software distributed database from data acqu. to telemetry handl.	3000 1.4 MB	16,000 24 MB	10 min
AOCS GNC	**Distributed synchronous systrem** 16 Processors inherent data buffering	838 24100	201,820 8.6 MB	10 min
	Operations on User-defined Types e.g. Little / Big Endian	341 6.8 KB	2300 42 KB	1 sec
	Interface adaption + On-Line Help Facility > 500 functions	1300 77 KB	145,00 3.4 MB	30 sec
	User's manual > 500 functions, 1200 pages, RTF	3400 425 KB formatoverhead	27,000 2.5 MB	1 min

Derivation of Man Power Equivalent

- Man Power Equivalent
  - equivalent man power needed to provide the automatically generated software by manual development
  - cost equivalent: (in-house) costs of equivalent man power
- Derivation
  - calibration of automatically generated lines, conversion to LOC
  - conversion of LOC into man power
- Calibration
  - take a reasonable figure on target bytes (executable code) per LOC
  - downsize the overall lines (incl. Scripts etc.) to LOC
- Conversion
  - take a reasonable figure on LOC / man hour

Target Budget for the Distributed Real-Time System

Instrument. Option	**CPU** Size of executable / MB gcc 386 gcc SPLC	**Net Lines** **Net Bytes**	**Bytes / Line** **and** **Source Bytes / Object Bytes**
full	4.7 -	400,000 17.3 MB	12 / 3.7 -
medium	3.9 3.2	380,000 15.8 MB	10 / 4 10 / 5
none	1.6 1.7	335,000 13.3 MB	5 / 8.3 5 / 7.8

Related Figures on System Generation

- System size
  - about 13.3 MB / 335,000 source lines without scripts etc.
  - about 1.6 MB on SPLC / VxWorks target
- Calibration (pessimistic)
  - 40 .. 50 bytes of target executable per LOC
    reasonable: 10 .. 20 bytes per LOC
  - 10 LOC per man hour (typical for space: 0.1 .. 2 LOC/h)
- „Pessimistic“ calculation of man power
  - factor 10: system size equivalent to about 40,000 LOC
  - 40,000 LOC equivalent to 4,000 man hours or 2.5 man years
- Productivity
  - an equivalent of 2.5 my is available within 20 min. on PC-800MHz
  - system configuration easily changable and maintainable
  - fast cycles of incremental refinement

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	**7.5**	**600 kEuro**
Data Processing and Database Software	10 minutes	5,000	500 man hours	**2**	**160 kEuro**
Distributed Synchronous System	10 minutes	20,000	1.25 man years	**7.5**	**600 kEuro**
Operations on User-defined Types	1 secomd	200	20 man hours	**> 36**	**> 3 Mio.Euro**
Interface adaption + On-Line Help Facility	30 seconds	15,000	1 man year	**120**	**10 Mio. Euro**

Quality

- 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

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

BSSE System and Software Engineering

Conclusions (1/2)

- Benefits of automation as proven by benchmarks
  - higher flexibility
    - every system configuration immediately available
    - an update is immediately available
  - shorter time-to-market and risk reduction
    - an equivalent of 2 .. 100 my available within 1 hour
  - increased productivity and quality
    - typical generation rates of 100,000 LOC per hour for correct systems, initial bug rate < 10^{-4} / LOC, this does NOT mean: 10 bugs / 100,000 LOC !!!!!!
    - manual implementation effort disappears
    - only system engineering effort remains

Conclusions (2/2)

- Status
  - current coverage of system domain
    - medium- to large-sized embedded systems
    - real-time systems, distributed systems, client-server systems
  - current coverage of the domain of functional software
    - data processing chain from acquisition to telemetry handling
    - command handling, distributed (on-board) database
    - interface adaptation, operations on user-defined types
  - inherent coverage
    - automated documentation (RTF, MS-Word, pdf)
  - but still far from full coverage of software domain

    however: the more experience, the more domains can be covered
- Future Work
  - automated GUI generation, automated re-engineering of legacy systems, DSP platforms possible

Final Remarks on “Software Crisis”

- Potential sources of the crisis
  - software production implies man power
  - higher quality implies more man power
  - missing benchmarks on the benefits of software methods and tools
  - $\rightarrow$ missing tuning of software development approaches

- Limitation of scope regarding problem solving
  - focusing on the experience of the past
  - current standards manifesting manual-oriented development

    $\rightarrow$ preventing a higher degree of automation
  - request for man-power intensive activities

    e.g. request of implementation-related documentation and reviews is not compliant with policy of an automated process chain at all

Organisation of Work Needed

- Frequent arguments against automation
  - ? our application is unique
  - ! re-organise it, surely it can be covered by automated construction
  - ? we need to know what happens
  - ! do we really know it in case of manual coding? (bugs)
    an automated production chain is completely known
  - ? we do not have „500" functions
  - ! re-organise your application, so that automation pays out
    it can be done, for sure
- Re-organisation example
  - manufacturing of electronic boards
  - in early days components did not have a shape well suited for automated leading and fitting
  - to allow for automation shape was changed towards SMD
  - lessons learned: be open for improvements