BSSE System and Software Engineering

Dr. Rainer Gerlich

ASaP

Automated Software Production

To Get a Software Product "As Soon as Possible"

ASaP applies the experience of other industrial areas to software development regarding the mastering of high complexity at reasonable costs and short time-to-market. By ASaP a minimum of human inputs - real creativity - are converted into the desired software product.

ASaP applies construction rules

- to transform literals and figures into an executable real-time / distributed system or other executable software, to automatically stimulate the system for verification and validation (including stress testing and fault injection), and to provide a report on the properties of the executed system,
- to extend already available software by automated generation of data and functions in order to complement the existing functionality.

  Hence, only some part of a software package needs to be generated manually, the other, possibly major part can be produced automatically. Also, glueing software interfacing with existing functions and data can be generated this way.

It was a major challenge to achieve this degree of automated software production for the following reasons:

- the right parametrisation and modularity had to be found

  in order to cover a wide application area and to avoid overhead,
- the process of automated software construction must yield software which is free of errors, executable and correct w.r.t. the provided inputs,

  which implies to reject erroneous, incomplete and inconsistent user inputs,
- the current platforms (operating systems, compilers, scripting languages, shells) do not support at all the fully automtated generation of such software without any human intervention after the generation process has started.

Therefore we had to perform a lot of iterations to continuously improve our concept and to make it feasible for industrial use. According to our knowledge, BSSE is the only company worldwide which supplies such a technology.

Below, we discuss a number of aspects related to automated software production and provide information on executed industrial projects.

Automation -

An Overloaded Term:

The terms "automat**ed** software production" and "automat**ic** code generation" address very different approaches.

"Automatic code generation" means to provide inputs e.g. on a graphical level from which the source code is automatically generated. In this context, the ratio between user inputs and outputs from the code generation tool is nearly 1 or close to 1, and a lot of manual intervention is still needed during the development process. Hence, this type of "automation" only covers a small part at the end of the development process and does not reduce the complexity the engineers have to deal with. Especially, the complexity and the risks of the first lifecycle phases are not reduced.

ASaP, the "Automated Software Production", covers the whole development process, requires only little information by the engineers and constructs the desired software product from a few inputs. It also covers verification and validation of the product's properties. An output/input ratio of more than 100 can easily be achieved. This does not mean an "overhead" of 10,000%, but that the gain is really 100.

Automation vs. Paperwork:

Additional procedures, standards, reviews and audits based on a lot of documentation are the usual answer to manage increasing complexity. However, this does not reduce the complexity of the system-under-development, but adds a lot of effort and costs. The amount of procedures which have to be followed increases and this neither makes the development process simpler nor better understandable.

At BSSE we found that automation is the right answer to master increasing complexity. What has already been mastered is taken out of the manual development process. A proven automated procedure takes the role of the engineer who can then concentrate better on the challenging issues.

Automation & Correctness:

The construction scheme, which is applied by ASaP, transforms proven software fragments into error-free / bug-free software. A wide range of applications and system structures are covered.

Software which is plugged into the generated framework or attached to ASaP software is checked for interface compliance. The result of automated production is software which is compliant with the user's inputs. If such inputs do not allow correct construction they are rejected. Vice versa, when the inputs are accepted an executable system is automatically generated. By execution the compliance with performance and resource constraints can immediately by checked.

Complexity:

Human beings can only handle a limited degree of complexity without failing. Especially, this is true for software development because we don't have sensors which tell us during development what a program really will do when it is operated.

To successfully enter a higher level of complexity we need to apply a new technology which reduces and limits the complexity an engineer is faced with. This has been well demonstrated by other areas like production of cars and electronic equipment. The answer given by ASaP is: *the new technology we need is automation of software production*. Of course, a number of ways towards automation exist. ASaP takes the way which allows to automatically construct software.

By introduction of a well-defined process model, which allows for automation, an increased complexity of the production process can be well mastered at a higher level of quality and precision.

ASaP organises software development such that automation can be applied to all phases of the development lifecycle.

Productivity:

A significant increase of productivity is possible if human intervention is limited to provision of inputs and to analysis of the achieved quality at the end of the production process. Any manual activities which are required during the development process itself decrease the overall productivity significantly.

Consequently, ASaP introduces an organisation scheme which allows full automation.

Quality:

A certain level of quality can only be achieved if the production follows a well-defined development process as ISO 9000 suggests it. If a lot of human intervention is required, quality cannot exceed the level of handcrafted products. But we all are looking for a guaranteed quality which can be improved stepwise at decreasing costs. Automation of the production process is the ideal answer to such issues.

Time-To-Market: A short development period can only be achieved if the production process is fully under control and iterations regarding optimisation do not require much time. A fully automated approach covers both needs.

Risk Management: In case of manual software production, the achieved properties of a software product often differ from the desired properties. As much time is needed until the first results are available, the chance to correct a problem earlier is very low. This imposes high risks on a project when the traditional development process is applied.

ASaP significantly reduces this risk because an executable system is available right from the first idea until completion of the final version. An engineer can check if the (first) ideas are feasible or not. No source code has to be written to get the first executable versions.

By incremental refinement the final version is smoothly approached, getting always a feedback from the actual version at any time. It is even possible to start integration right from the beginning.

Cost Assessment: When tailoring ASaP towards a certain application we identify the main cost drivers quantitatively.

Usually, only qualitative cost assessments are done like: "use of programming language A is x % more expensive than in case of B", "if you apply standards STD1 it costs more than for STD2", or "for this type of application a productivity figure of PF1 may be achieved while for this type it may be PF2". This does not help so much regarding cost minimisation for a specific application domain.

In case of ASaP we identify which software can automatically be generated when certain inputs are available. Therefore only the absolute minimum of information is required to generate a software product automatically.

This way the dependency on the amount of items which drive the costs like nodes of a network, functions, messages etc. can be reduced from a high-order dependency to a low-order dependency. This leads to immediate and measurable cost savings. As costs are mostly related to working time, the development time will be shortened, too.

In our projects we could reduce such dependencies e.g. from $F(x)=Ax^2 + Bx + C$ to $f(x)=ax + b$ where $a << A$ and x was in the range of about 20 nodes, or from $F(x)=Ax + B$ to $f(x)=a$ where x was in the range of about 300 functions.

Due to the scalability of the ASaP approach no re-testing or additional testing of already existing software is required in case x varies or increases.

To achieve a maximum of cost saving it is important to find the right entry point. Product development starts with little information which steadily grows during design, coding and testing. Therefore we start with automation at a point where only little information has to be provided manually, i.e. as early as possible. For each new application we identify the optimum entry for ASaP according to the information we receive from our customer.

Resource Optimisation: ASaP removes a lot of workload from engineers which they need to spend for coding, testing and integration in case of the traditional, manual approach. Edison mentionned that for an invention "1% inspiration and 99% transpiration" are required. While in this case the "transpiration" cannot be removed, ASaP does remove it as much as possible from the software production process by proper organisation.

Consequently, much time is saved which the engineers can spent on "inspiration", i.e. they can concentrate more on a product's properties at a rather high productivity rate.

Portability:

ASaP supports "transparent" distribution which implies full portability across the supported platforms. Each part of a software product may run on any of such platforms, vice versa, the software product may also run on a platform subset. If the topology is changed or platforms are replaced, only a few bytes have to be changed per platform.

If the product is supposed to run on N nodes, the system is executable on any topology between 1 .. N nodes. This gives high flexibility for integration because the system can be pre-integrated on a smaller and possibly different network, and it will immediately run on any other and - of course - on the final configuration.

Automation vs. Reuse:

Automation provides the capability to flexibly adapt to different, possibly floating requirements without any need for human intervention other than initial inputs. Therefore systems of different structure, e.g. an embedded real-time system or a client-server system, can be constructed by the same generic generator without including overhead.

Reusable software - as it is understood today - cannot cover structural changes e.g. of Finite State Machines or data types, without manual intervention. This also applies to classes as introduced by the object-oriented methodology in order to enforce development of reusable software.

Due to automation we can enter a higher level of abstraction which allows us to construct the classes automatically and to cover a broad range of application domains.

Verification &Validation:

For V&V ASaP takes a system-oriented position in order to succeed in practical cases. By construction it limits the state space. It does not produce a hay stack around a needle and then apply sophisticated methods and tools for identification of the needle within the hay stack, it just constructs the needle.

History:

ASaP is the outcome of activities which were started by beginning of '90s in the context of ESA (European Space Agency) projects aiming to define a new software development methodology. Since 1996 BSSE continued these activities by own projects - in part supported by ESA - and improved the actual approach stepwise based on the feedback from its projects.

In 1999 BSSE was ready to provide the first environment which fully automates the development process in the area of real-time and/or distributed systems. This environment is called "ISG" (Instantaneous System and Software Generation"). Since summer 2000 it is already in industrial use for the MSL ("Material Science Laboratory") project which is an experiment on-board of the International Space Station (ISS). ISG is capable to provide the complete real-time and communication infrastructure from inputs provided e.g. by a formsheet as literals and figures.

In the context of MSL BSSE also built ASaP software which automatically generates the database software including the database structure itself, the monitoring, calibration, data acquisition, telemetry and command handling software.

By follow-on projects BSSE learned how to build automatically specific application software and improved and enhanced this technology from application to application.

Today, BSSE has the experience, the knowledge and the tools to flexibly react on a customer's request for automated software construction.

To base such an automated approach on commerical software which is on the market is still a challenge because such software is not designed for automated operation at all. However, due to the long experience of BSSE such challenges have been mastered.

What ISG Does:

ISG covers the domain of real-time and distributed systems. ISG was the first step regarding full automation. By ISG we collected the experience which allowed to enter the next step: ASaP.

ISG (Instantaneous System and Software Generation) automatically generates the real-time and communication infrastructure from e.g. spreadsheet inputs. ISG provides generic interfaces to plug-in other software which is either also generated automatically, which already exists or is generated by other tools.

ISG not only generates software, but also distributes it automatically across a network of nodes. Each such node may be based on a different processor type or operating system. Moreover, ISG provides the means for automated verification and validation of the software product, including the capability for automated instrumentation, stress testing fault injection, data logging and evaluation, graphical presentation of data flow and events.

What ASaP Does:

ASaP reflects BSSE's experience in organising the development of a software product such that a maximum of automation is achieved.

Due to automation the costs, time-to-market and risks are decreased and the quality is increased.

ASaP addresses the whole scope of automated software construction, it includes ISG as a subset.

What BSSE Already Did:

Since a number of years BSSE is applying and improving the ASaP technology.

ISG

ISG has successfully been applied to the MSL project and a lot of on-board software has automatically been constructed (see above for more details). ISG generates the MSL real-time infrastructure as defined by a spreadsheet (about 40 process types, about 50 process instances, two nodes, distributed database) within a period of about 20 minutes from scratch (on a PC-800 MHz platform).

ISG has been used to investigate the impact of faults and time jitter in case of a distributed synchronous voting system consisting of 16 nodes (ESPRIT project CRISYS). As a result the sensitivity against certain faults and time jitter was identified. The CRISYS infrastructure (16 processes instances, 9 process types, 16 nodes) was generated within a period of about 10 minutes from scratch (on a PC-800 MHz platform).

In this context the ISG infrastructure has been integrated with Scade software. Scade is a tool for generation and verification of synchronous systems. Scade components have been plugged into the distributed framework as generated by ISG.

ISG allows to provide performance characterisitcs independent of the network topology. E.g. about 270 figures are needed to specify in detail the transfer rates of the node-internal channels and the channels which

connect a network of 16 nodes. If the channels have the same performance characteristics, only 3 figures have to be provided. If the number of nodes, their logical names or their (IP-)addresses are changed, no modification of these inputs is needed.

Moreover, ISG has been used to model parts of a mail sorting and distribution machine in the context of CRISYS.

ASaP ***"Invest once, gain forever"***

Functions for arbitrary operations on any user-defined data type

We had to exchange binary data between processors based on different memory architectures ("Little Endian", "Big Endian") for a number of complex user-defined data structures. Therefore we automated the construction of the conversion functions. We only need the definition of the relevant data structures (e.g. provided by h-files), then we press a button, and within less than 1 minute we get the needed conversion functions for whatever data types.

The same procedure is applied to initialise any user-defined datatypes with random or pre-defined values and to evaluate the results.

By this ASaP software we can easily cover any other operations on data types.

Building interfaces

By ASaP we built an interface between C libraries (about 300 private functions plus C standard libraries) and a scripting language. This task included stack alignment of parameters, description of the parameters, if needed adaption of types, embedding of the library functions into scripts and provision of context-dependent parameter values for such scripts. Moreover, we could easily provide an "executable help" facility by which a user can exploit on-line the capability of a library function without any need to generate an own script.

Whenever a new function was added, the interface was automatically adapted by the ASaP software. No tests were needed for integration of the new functions into the scripting environment.

All the files needed for about 300 functions are built within less than ½ minute (on a PC-800 MHz platform).

Converting data into executable scripts

We consider spreadsheets (tables) as a compact and comprehensive form of a data representation which is easy to maintain. To feed a number of data into a GUI (Graphical User Interface) application, e.g. a browser or database interface, time-tagged or event-driven, we converted spreadsheet data into executable scripts which operate the GUIs. Hence, an engineer does not need to learn the scripting language. Even if (s)he is familiar with it, the effort of writing and testing the scripts is saved.

Similarly, we use tables to define sequences of actions other than to feed in data, e.g. to operate and test GUIs, and generate automatically executable scripts from such data. The same advantages apply in this case, too.

To generate a script from a large table takes less than ½ minute (on a PC-800 MHz platform).

Building a database and related acquisition, calibration, limit monitoring, data processing and telemetry functions

ASaP generates the MSL distributed database within about 10 minutes from scratch (PC-800 MHz platform). This software complements the MSL infrastructure as generated by ISG.

The MSL database had to be restructured to solve a performance problem (CPU load close to 100%). This could be done just be changing the contents of a column in the spreadsheet which defines the grouping of the data items and the database structure. The restructured database was generated immediately, no further testing was needed, because all the interfaces of the MSL software were automatically updated in a consistent manner.

The manual approach would (possibly) have required about 1 man month (or even more time), because about 600 data items had to be re-grouped and the related functions (data acquisition, calibration, limit monitoring, telemetry fram generation) to be changed and tested. Moreover, the success of the change would have been unknown for such a long time, while in case of ISG/ASaP the result was immediately available.

What Can Be Done:

In addition to above application areas ASaP may be applied to other domains like databases and GUIs, too. Typical (generic) application areas are the adaption of interfaces and software integration. Our experience is that we can provide solutions for a lot of domains.To find a solution we are putting together our experience with the experience of our customers.

The use of ASaP and ISG is not limited to the technical domain. E.g. simulation of business or logistic processes could be covered as well

We leave it up to potential customers to identify more application areas together with us.

Generation Time:

The generation time as given above apply to any systems of similar complexity.

In case of ISG it depends on the number of process types, number of different platforms and the amount of exchanged messages.

In case of ASaP it depends on the number of relevant components.

The observed dependency is approximately linear plus a fixed offset.

Availability:

ISG and ASaP have been applied already successfully to industrial projects.

ISG is available for Un*x (Solaris, Linux) and VxWorks and PC and Sparc in every possible combination of software and hardware platforms.

The complementary ASaP software is platform independent - in principle, but this actually depends on the given requirements. E.g. some parts of the software mentionned above has been ported from Un*x to Mac OS 9, from gcc to MPW and CodeWarrior compilers without problems. If no specific interfaces need to be covered, the ASaP software is portable in general, if specific interfaces have to be met on customer's request, it may be bounded to a certain platform.

Benefits:

Within minutes ISG converts ideas into a real executable system which generates a report on the system's properties. This remains true even for large and complex systems.

ASaP reduces the effort by one order of magnitude at least regarding the cost driving elements.

Following the idea of ASaP only a part of a software product needs to be built manually, the other parts will be constructed automatically based on information derived from the already existing parts.

This way e.g. interfaces can be established automatically. It is also possible to generate MMIs, help facilities or training software for a product.

How to Take Advantage:

BSSE applies ISG and ASaP in-house to build software products and turn-key systems according to a customer's specification.

BSSE provides licenses on ISG and training so that a customer can build his own products.

BSSE provides tailored ASaP tools on request.

BSSE provides support and training to implement ASaP in a customer's organisation for process improvement.

Summary:

Automation is - in our believe - the only answer to master the increasing complexity in the area of system and software development.

Automation requires an appropriate organisation of the work to be done. The traditional organisation schemes foresee continuous human intervention during the production process. This turns out as counterproductive and prevents any significant progress towards higher productivity and mastering of higher complexity.

The use of the right organisation scheme is a pre-condition for future success. BSSE has needed about a decade and a lot of iterations to find a way for efficient organsiation of the software development process based on automation.

ISG takes only user directives and automatically converts them into a distributed executable system, thereby silently providing means for operational and stress testing, fault injection and evaluation of the system's properties.

ASaP takes existing software, possibly specifcations only like date type definitions and function prototypes, and user directives. From such inputs it automatically produces more software which interfaces with or complements the already available software. This way a major part of the total software may be established easily.

By ISG and ASaP a customer can get an immediate and measurable return of his investment and better and easier master future challenges.

ISG and ASaP are not just ideas, they have already been applied to industrial applications, and their benefit is measurable.

Outlook:

ASaP and ISG are evolving technologies which now have become sufficiently mature for external use, after BSSE applied them for a long time in-house. According to the feedback from projects BSSE is steadily improving and extending this technology.

Publications:

A number of papers on ASaP, ISG and their roots are available on request.

Point of Contact: Dr. Rainer Gerlich BSSE System and Software Engineering
Auf dem Ruhbuehl 181
88090 Immenstaad, Germany

Voice: +49/7545/91.12.58
Fax: +49/7545/91.12.40
Mobile: +49/171/80.20.659

e-mail: gerlich@t-online.de
URL: http://home.t-online.de/home/gerlich