BSSE System and Software Engineering

Avoiding Malfunctions Due To Software Failures by Automation of Software Production and Test

Pannen wegen Software-Fehlern vermeiden durch Automatisierung von Software-Produktion und Test

ESA Colloquium
„Technology Exchange between Space and Automotive Industry“

ESOC, Darmstadt, Germany 15.10.2002

Detailed Presentation

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

Facts and Issues

- Wirtschaftswoche summer / autumn 2002
  - increasing number of malfunctions in high-tech vehicles
  - major part by software malfunctions
- VDI-Nachrichten September 2002
  - Charles Simonyi, co-founder of Microsoft

  „today we can design and control production of jumbo-jets, but programming still remains a handcrafted task“
- Trend in automotive industry: more software
  - e.g. 11. Aachener Kolloqium „Fahrzeug- und Motorentechnik“
  - high quality software becomes a big challenge

Looking on Car Mass Production

- well-defined production process
  - high productivity
  - high quality
  - pre-condition for quality assurance
  - saving company know-how
  - continuous process improvement
- automated software production becomes "the" issue
  - scalable approach needed to cover a broad range

BSSE's Automated Approach ASaP

History

- 1992 ESA project on performance and FDIR validation (HRDMS)
  FDIR = Fault Identification and Recovery
- 1993 ESA project on behavioural validation (OMBSIM)
- 1995 ESA project on sysem fault tolerance (DDV)
- 1996 BSSE project on ATC protocol validation (OPAL)
- 1997 BSSE project on distributed fault-tolerant system (CADIS)
  turn-key system
- 1998 BSSE project on distributed critical control system (CRISYS)

ASaP milestones

- 1999 first version of fully automated environment (MSL / ISS)
- 2000 final delivery of automatically generated software package

ASaP Improvements

- State-Of-The-Art

• Productivity:	1 man-hour	0.1 .. 10 LOC
• Bug Rate	typical:	10^{-2}/ LOC
	very good:	10^{-3}/ LOC

N.E.Fenton, 2000

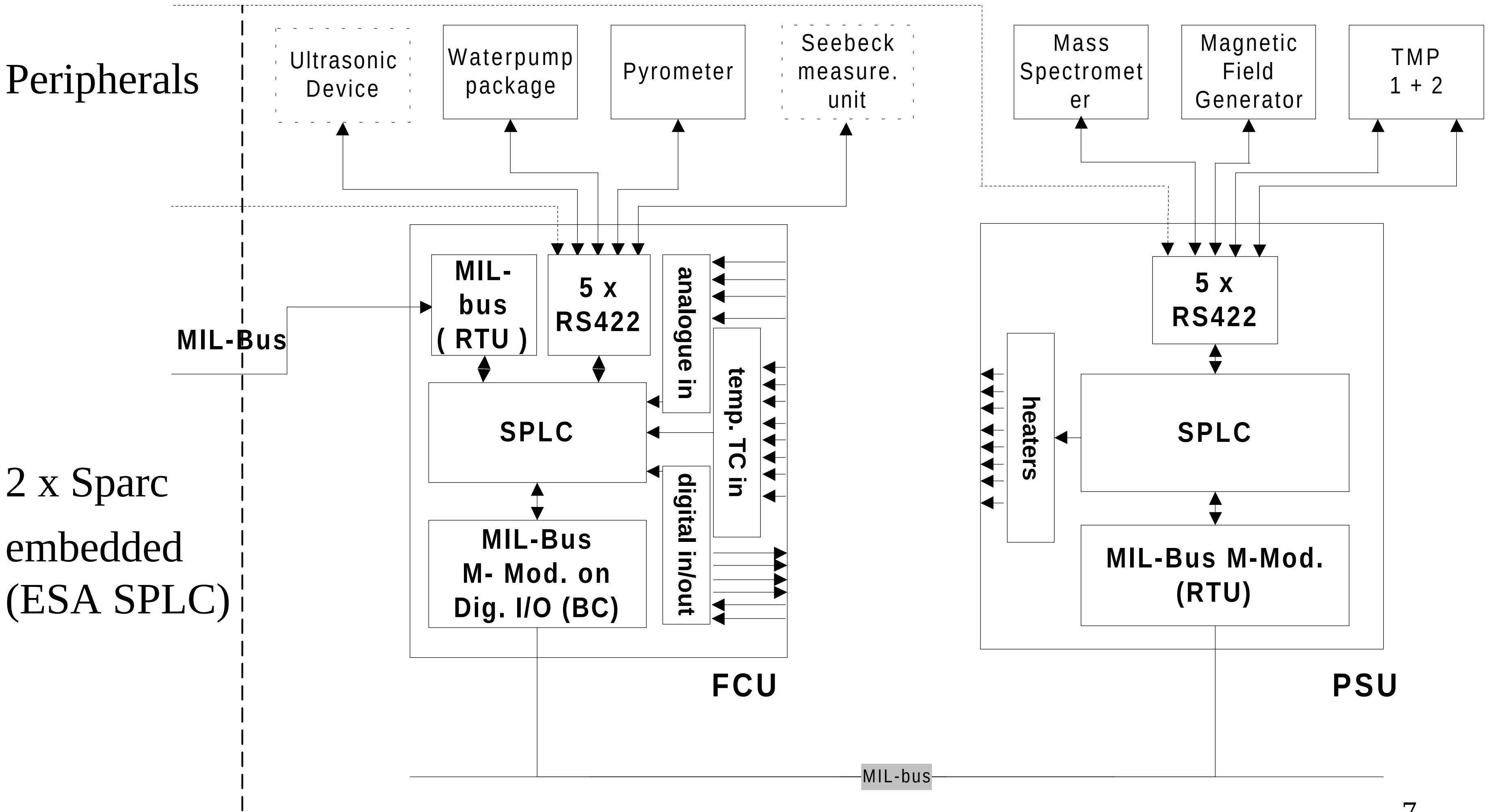
- Automated Software Production and Test (ASaP)

• Productivity	1 PC-hour	16,000 .. 320,000 LOC
		1 .. 20 man-years (my)
		1600 mh/my 10 LOC / mh
• Bug Rate		≈ 0 .. 10^{-5} / LOC

Quality is not expensive

if a new technology is applied

Example: Distributed Real-Time System (MSL / ISS)



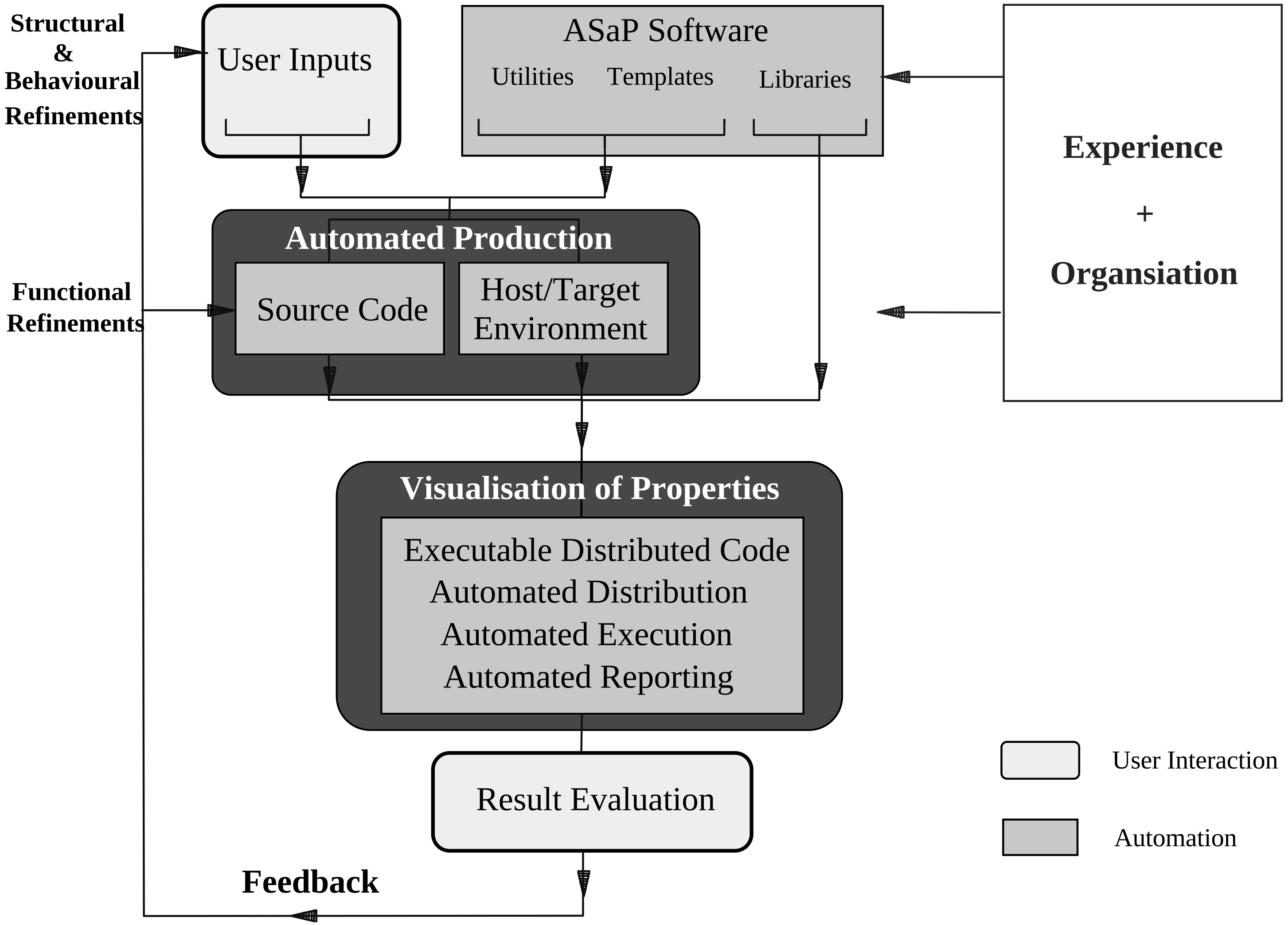
Synergy by ASaP

- Current status
  - much support of paper work
  - but little support to verify code
- we do not have sensors for software bugs, but ...
  - we do only concentrate on automatic code generation
  - we still apply manual verification and testing
- ASaP
  - synergy between generation and verification / testing by automation
  - less complex user interface
  - automated visualisation of properties

ASaP Process Improvement

- Complexity Reduction
  - adequate user notation
  - automated handling of dependencies
- Productivity
  - identification of high manual effort $\rightarrow$ automation
  - fully automated process chain
- Quality
  - less effort, less bugs
  - less complexity, less bugs
- Risk Reduction
  - immediate, early feedback on properties

Incremental Generic Development Cycle of ASaP



Evolving Automotive Applications

- Higher functionality and user comfort by software
  - higher intelligence of units

    fail-stop (Servo) $\rightarrow$ fail-safe $\rightarrow$ fail-operational (drive-by-wire)
  - infotainment
  - vehicle network
  - system - subsystem hierarchy

- Requirements on software
  - higher flexibility to implement a variety of functions
  - shorter development cycle
  - high dependability
  - good usability

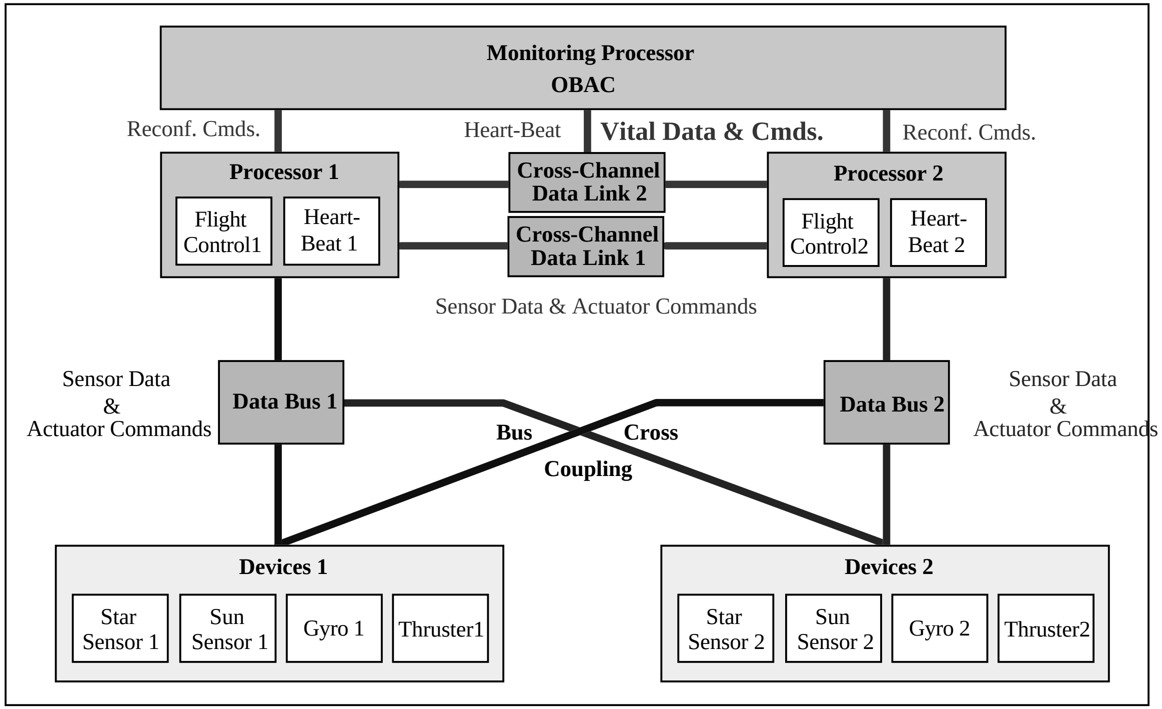
Benefits by ASaP

- flexibility for changes and refinements
  - generic approach
  - compact user inputs
- shorter development cycle
  - short generation time
  - incremental refinements
- higher dependability
  - fault prevention by intensive checks of user inputs
  - fault identification by built-in assertions and visualisation
  - fault tolerance by coverage of exceptions, built-in mechanisms
- higher usability
  - fast feedback from real system
  - little effort to optimise the user interface

Potential Application Areas in Automotive Industry

- Vehicle
  - system management
  - subsystem / unit software
- User Interface / Infotainment
  - user operation
  - unit manangement
  - unit software
- Vehicle Manufacturing
  - automated verification of manually coded robot programs vs. CAD
  - automated generation of robot programs from CAD data
- Project and Contractor Management
  - integration of COTS and sub-contractor software
  - quality checks and evaluation of software

Fault-Tolerant Systems and Risks



ESA Project DDV

validation of the exception handling of a fault-tolerant system

Sporadic System Failure

- computer C1 fails
- computer C2 takes over control
- C1 issues an actuator command before it fails
- C2 does not get this information
- C2 issues the same command
- the system fails

Risk Reduction by Automation

- Can such problems be identified in advance?
  - by a systematic approach
  - by spending more time on system engineering

- Answer by automated software production: YES
  - framework implies problem classification
  - less time needed for implementation,
    more time available for system engineering
  - better visualisation of properties

Executable Specifications

- ASaP
  - executable system right from the beginning
  - system interfaces are already defined by „stubs“
  - „executable specifications“
- Subcontractor Management
  - take executable specifications as reference
  - distribute full executable environment
  - each subcontractor gets a reference unit
  - each subcontractor can pre-integrate its units
- Integration on next higher level
  - replace stubs by subcontractor deliverables
  - verify and validate the actual version

Inputs in User Notation and Derived Output (MSL Database)

Name of Signal	Data Type	Input Range	Physical Range	Acqui. Rate	HW Module	Calibration Type
CFDdrive_pot	REAL32	0 - 10V	0 - 200 mm	100	ASM F1	FctASM1_Std
CFDrot_pot1	REAL32	0 - 10V	0 - 360 °	100	ASM F1	FctASM1_Std
CFDrot_pot2	REAL32	0 - 10V	0 - 360 °	100	ASM F1	FctASM1_Std
CF_reg_v_pot	REAL32	0 - 10V	0 - 270 °	10	ASM F1	FctASM1_Std
GS_press_low	REAL32	0 - 10 V	0 - 2 bar abs.	10	ASM F1	FctASM1_Std
CFVpenn_chamb	REAL32	0 - 10 V	1.e-7 - 1000 mbar	1	ASM F1	FctASM1_Pressure
VGSpenning_ms	REAL32	0 - 10 V	1.e-7 - 1000 mbar	1	ASM F1	FctASM1_Pressure

```
T_database_entry MSL_db_desc[]={
                                        {
/* address in DB            */                    (int*)&MSL_db.LRT_HK_A1.CFDdrive_pot,
/* offset  in DB            */                    (int)CFDdrive_potDBoff,
/* #samples                 */                    100,
/* size of data type        */                    sizeof(REAL32),
/* id of type               */                    7,
/* copy DB data             */                    0,
/* calibration function     */                    {(int*)FctASM1_Std_CFDdrive_pot,
/* supervision structure    */                              {
/*   SV function            */                                        (int*)&limChckREAL32,
/*   limit definitions      */                                        CFDdrive_pot_suarr,
                                                            }
                                                  },
/* post-processing function */                    {(int*)NULL}
                                        },
```

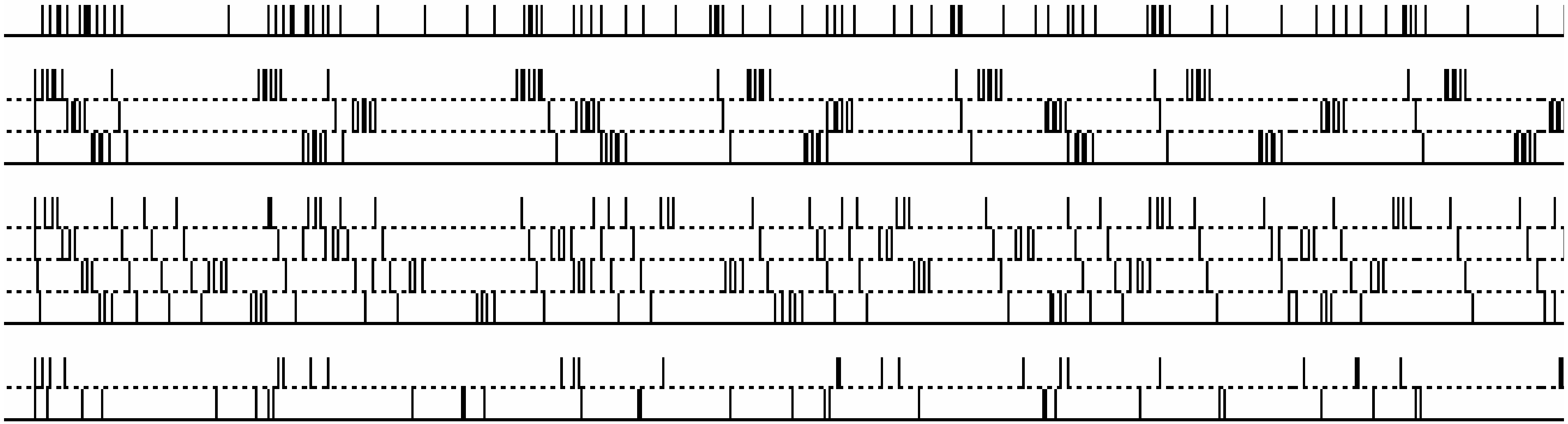
More Derived Files (Subset Only)

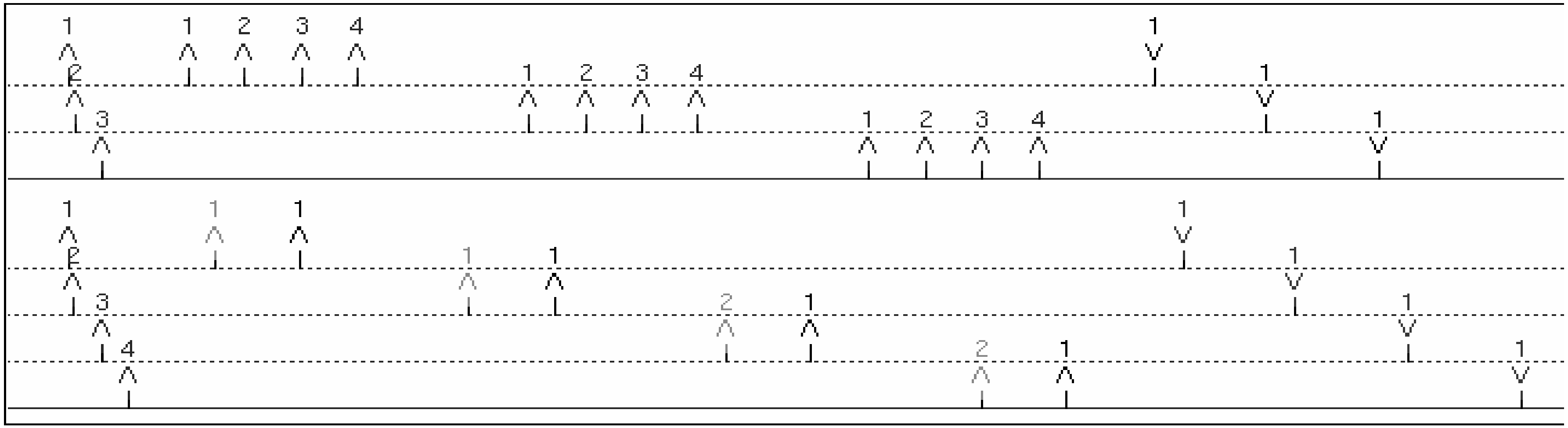
```
REAL32 FctASM1_Std_CFDdrive_pot(UINT16 Data)
{
          REAL32 buffer;
          INT32  value;

/* ADC correction (including amplifier correction) */
          buffer = (Data - ADCCorrASM1[0][0].Offset) * ADCCorrASM1[0][0].Gain;

/* conversion to physical units */
          value = (REAL32) ((REAL32)0) + ((REAL32) 200  - (REAL32) ((REAL32)0)) /
                    ( (REAL32) ((1<<12) - 1) ) * buffer;
          return value;
}
```

```
#include "frametypesDB.h"


/* MSL database */
TyDatabase MSL_db;

/* Recording of database updates */
int updateDBcnt=0;
int DBupdate[TOT_DATABASE_ITEMS];

/* # of instances of telemetry frames */
#define TM_BUFF_INST 2
```

```
int LRT_HS_framePtr=0;
int LRT_HS_frameInd=0;
TyTMbuffer_LRT_HS LRT_HSArr[TM_BUFF_INST];

/* Array pointing to TM frame instances */
TyTMbufferArr TMbufferArr[]=
{
          {LRT_HK_A1_ID,      (int*)&LRT_HK_A1Arr},
          {LRT_HK_D1_ID,      (int*)&LRT_HK_D1Arr},
          {LRT_HK_D2_ID,      (int*)&LRT_HK_D2Arr},
          {LRT_HS_ID,         (int*)&LRT_HSArr}
};
#define TM_BUFFER_ARR_SIZE
          sizeof(TMbufferArr)/sizeof(TyTMbufferArr)
```

Visualisation of Properties: Timing and Communication





Reports available after 15 minutes starting by delivery of user's spreadsheet inputs

Achievements

- distributed real-time system
  - within 30 minutes equivalent of about 5 man-years (my)
  - 80,000 LOC (environment: 200,000 LOC)
- distributed, synchronised database
  - within 30 minutes equivalent of about 1 my
  - 16,000 LOC and more
- operations on data types, interfaces etc.
  - within 1 minute equivalent of about 2 my

We give warranty!

"accepted user inputs are automatically transformed into correct and immediately executable software when applying an automated production process established by us"

How and Where Can ASaP be Applied?

- Use of existing products
  - distributed critical real-time and control systems
  - data processing, distributed databases, GUIs
  - test case generation
  - integration and subcontractor management
  - user support possible
- Customised ASaP approach
  - know-how transfer
  - definition of an appropriate approach
  - building of the needed environment

Customising ASaP

- Analysis of current manual procedures
  - similar to „REFA“ in case of hardware
  - identification of the most generic approach
    maximum coverage of application area
- Definition of the user interface
  - identification of driving parameters
    minimum set of user inputs
  - re-use of current environment (if any)
    building of an interface to the user‘s world
- continuous optimisation of ASaP procedures
  - provision of analysis tools
  - continuous benchmarking to check productivity and quality
  - continuous process improvement

This was a very short introduction, but ...

we are available today

- for further discussions
- to show more details
- to give a demo on

„Instantaneous System and Software Generation“ (ISG)

15 minutes from user's spreadsheet inputs to reports

- **10 minutes generation time**
- **3 minutes system execution**
- **2 minutes report generation**