Lecture Notes in Networks and Systems 674

Herwig Unger
Marcel Schaible *Editors*

Real-time and Autonomous Systems 2022

Automation in Everyday Life

Springer

Lecture Notes in Networks and Systems 674

The series "Lecture Notes in Networks and Systems" publishes the latest developments in Networks and Systems—quickly, informally and with high quality. Original research reported in proceedings and post-proceedings represents the core of LNNS.

Volumes published in LNNS embrace all aspects and subfields of, as well as new challenges in, Networks and Systems.

The series contains proceedings and edited volumes in systems and networks, spanning the areas of Cyber-Physical Systems, Autonomous Systems, Sensor Networks, Control Systems, Energy Systems, Automotive Systems, Biological Systems, Vehicular Networking and Connected Vehicles, Aerospace Systems, Automation, Manufacturing, Smart Grids, Nonlinear Systems, Power Systems, Robotics, Social Systems, Economic Systems and other. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution and exposure which enable both a wide and rapid dissemination of research output.

The series covers the theory, applications, and perspectives on the state of the art and future developments relevant to systems and networks, decision making, control, complex processes and related areas, as embedded in the fields of interdisciplinary and applied sciences, engineering, computer science, physics, economics, social, and life sciences, as well as the paradigms and methodologies behind them.

Indexed by SCOPUS, INSPEC, WTI Frankfurt eG, zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

For proposals from Asia please contact Aninda Bose (aninda.bose@springer.com).

Herwig Unger · Marcel Schaible
Editors

Real-time and Autonomous Systems 2022

Automation in Everyday Life

Springer

Editors
Herwig Unger
Lehrgebiet Kommunikationsnetze
FernUniversität in Hagen
Hagen, Germany

Marcel Schaible
Lehrgebiet Kommunikationsnetze
FernUniversität in Hagen
Hagen, Germany

ISSN 2367-3370 ISSN 2367-3389 (electronic)
Lecture Notes in Networks and Systems
ISBN 978-3-031-32699-8 ISBN 978-3-031-32700-1 (eBook)
https://doi.org/10.1007/978-3-031-32700-1

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The present volume is unique in several ways and, in a certain sense, also represents a new beginning. For the first time, readers will find the contributions to the conference 'Echtzeit 2023 (Real-Time Systems 2023)' and selected papers from the international conference 'Autonomous Systems' in one volume, and also for the first time, the contributions to the German-language conference 'Echtzeit' are published in English.

Both conferences, organized by the Real-Time Systems Group of the German Informatics Society, were held pandemic exclusively in attendance in 2023 for the first time since the COVID-19 pandemic period. In both cases, the organizers fight for old and new audiences and authors, in both cases they try to help new scientific findings to spread in the scientific community and last but not least wanted to support the building and maintenance of networks of scientists through a variety of discussions and new contact opportunities. These concerns are even more difficult to realize in the post-pandemic era, in particular through the permanent presence of online meeting systems and newly built electronic community systems.

It is only logical that the partly joint organizers of both conferences try to achieve this task by being more effective and by joining forces. The present volume realizes the demand of the younger generation of scientists to present results in real-time research in English and thus reach a wider audience and, with the transition to Springer as publisher, a more international audience is created for the contributions to the topic of autonomous systems. For both conferences, the now joint conference proceedings means both a reduction in work and costs and the possibility of getting more papers in hand to read without more effort.

The first part of the present volume comprises all contributions to the conference 'Real-Time Systems 2023', traditionally held in Boppard on the Rhine. The series opens with Peter Holleczek's contribution to the 'Real-Time Archive', which he has been building for many years as one of the first participants with a huge number of experiences in relevant research. His unique review of developments, especially of the programming language PEARL, was an excellent opening of our conference. The winner of the student competition 2023, Mr Jan Knoblauch, shows in his paper on deadlock detection in PEARL that the research direction pursued since the 1970s is still relevant today and can inspire young researchers.

In the following contributions, authors have their say on many aspects of real-time systems and hardware, especially in industrial systems. The number of contributions submitted under the topic of safety and security proves the increasingly important role of security aspects, especially for real-time systems in the daily use, which are—in particular—also considered, when designing software with PEARL. Presentations on current applications round off the conference program. The second part of the volume contains selected papers from the international conference and doctoral seminar 'Autonomous Systems', which has been held by the working group 'Real-Time Communication' in

Mallorca at the end of October almost each year since 2008. Although the number of participants at the conference is much larger, the number of contributions is much smaller, since many participants of the conference come to Mallorca to make new contacts and discuss new ideas in detail, which also inspires young scientists and doctoral students. In addition to Janusz Kacprzyk, Zhong Li and Peter Kropf, Stefan Pareigis was one of the keynote speakers who was present at both conferences and whose therefore extensive contribution represents a combination of both events. With his topic 'Autonomous Driving', Stefan Pareigis also shows new and common challenges for both real-time and autonomous systems at the same time. An interesting combination of contributions of topics such as machine learning, fault tolerance and decentralized systems rounds off the presentations, in which, of course, innovative ideas on managing the flood of information in the WWW of tomorrow must not be missing.

The editors of this book 'Advances on Real-Time and Autonomous Systems' hope that you will enjoy reading this book that you might be able to experience a bit of the inspiring moments of both conferences and, finally, that you may get a few own ideas by reading it.

Of course, we would be pleased to welcome one or the other of you at one of our conferences in the next year, maybe also as an author of a contribution. For the conference 'Autonomous Systems', which will take place from October 22 until 27, again in Cala Millor in Spain, all information is available at https://www.confautsys.org, while the 'Real-Time Systems' event traditionally takes place again on November 15 and 16 in Boppard on the Rhine and everything important can be found at the web address https://real-time.de/tagungen.

January 2023

Herwig Unger
Marcel Schaible

Contents

Real-Time

Real-Time

The Real-Time Systems Archive

Peter Holleczek(✉)

Friedrich-Alexander University of Erlangen-Nuremberg, National High Performance Computing Centre (NHR@FAU), Martensstr. 1, 91958 Erlangen, Germany
peter.holleczek@fau.de

Abstract. Having survived the end of the "paper" era, historically quite arbitrary documents from the inventory of the real-time systems area's founding fathers are currently digitised and included in a subarchive of the general archive `dl.gi.de` maintained by the German Computer Society (GI). This real-time systems archive is structured like a tree. Deep down, it branches out into subareas, each of which contains an explanatory summary. Contributions usually include their full text, an abstract and a citation. Proceedings are represented both entirely and by individual contributions. In its current form, the archive shows the enthusiasm of the developers, and also economically successful implementations of real-time topics in the 1970s and 1980s. Obviously, this GI subarchive has arrived on the scene as search engines now prefer it when searching for real-time topics. Its content and how to deal with it are presented below.

Keywords: Real-time systems · real-time programming languages · process control computing

1 Introduction

Real-time systems, like IT as a whole, were born in the "analogue" world. While it is said that the IT product internet does not forget anything, the analogue documents of the early days are threatened with the "analogue end". The founders of the real-time community are retired by now or have left us completely. Their bookcases have been cleared. Content was preserved more by chance, the compilation of which was rather random. As most of the basic considerations of that time are still valid today, however, it would be good to have an archive saving what can be saved.

2 Real-Time Systems Within the GI Archive

In Germany, the real-time community is organised in the specialist committee "Fachausschuss Echtzeitsysteme" within Gesellschaft für Informatik (GI). The core of this community can be traced back to the development of the real-time programming language PEARL in the 1970s, but has, long since, opened up

H. Unger and M. Schaible (Eds.): Real-Time 2022, LNNS 674, pp. 3–12, 2023.
https://doi.org/10.1007/978-3-031-32700-1_1

to real-time systems in general. Fortunately, GI had recognised the problem of "forgetting" knowledge related to computer science and, as a remedy, set up a digital archive (https://dl.gi.de), which is open to all specialist groups, thus following the example of ACM (https://dl.acm.org).

The GI archive is particularly dedicated to "grey" literature, which, apart from scientific publications, opens up content important for practitioners. The portal has a German language interface, only. Naturally, it contains many documents written in German, especially those from the development phase. As the GI archive covers many topics, it has a tree structure leading over several levels to the particular areas such as the one on real-time systems. So far, the contributions archived there are largely limited to historical documents up to the end of the 1990s. This content is to be understood from its historical context.

3 Historical Background

The real-time systems archive essentially breathes the zeitgeist of the 1970s and 1980s. Computers could be used for the first time as control tools for technical processes, both in the industrial and scientific world. Completely new dimensions opened up for users, even if the computers were room-sized in the beginning. Compared to discrete circuits, for example, their flexibility was essential. The process control computers of that time were initially adaptations of classic computers for sequential tasks. They were extended to include options to react to external events (interrupts) and to control actuating variables. What was lacking was suitable programming and associated programming systems. Soon it became clear that assembly language programming was out of the question for complex contexts. A new way of thinking in parallel cyclic processes, synchronisation, selected error handling, and interrupt reactions was appropriate. It was the heyday of international real-time language development.

In Germany, the Federal Government recognised the need for and the opportunities of generous public funding, manifested in the Federal Government's data processing programmes [6] at the beginning of the 1970s. Along with this came the call for standardisation, both nationally and internationally. Funding beneficiaries were the rising and globally active electrical industry, industrial outfitters, computer manufacturers as well as the scientific world with its emerging information technology and computer science scenes. One target of funding at this time was the implementation of the *P*rocess and *E*xperiment *A*utomation *R*ealtime *L*anguage (PEARL).

4 Description of the Archive

Currently, the real-time systems archive contains some 600 items.

4.1 Inventory

The starting stock was mainly fed from the more or less preserved paper collections of the start-up scene. PEARL inevitably appears in various places, and can be used here as an example. The inventory includes everything that was used at the time for language development, implementation and application, constantly with an eye on developments elsewhere. It extends in time to the entry into the digital age with desktop publishing and internet presence at the end of the 1990s. Stricter publication guidelines make it difficult to place them all in an open full-text archive.

4.2 Digitisation

Wherever possible, the documents were disassembled and scanned page by page, with a resolution of usually 600 dpi. Text recognition was turned on provided the font was reliably recognised and the layout was not garbled. Unique items, in which public libraries were still interested, were preserved as a whole and carefully scanned as books.

4.3 Structuring

Ultimately, the ad hoc-like inventory entails a pragmatic and not necessarily orthogonal structure. The archive's incremental growth leads in places to a terminology that one would not have chosen from statu nascendi. Now it is better not to change the terms that have been introduced. Inevitable latecomers may need to be matched. The archive is currently divided into 13 areas ("Bereiche", Fig. 1) according to historical arrangements.

Fig. 1. The archive's topical areas

For each area the overview in Fig. 1 shows the number of documents contained. The identifier of each area leads to an explanatory header at the next level and, usually, to further subdivisions. At the lowest level one finds the digitised objects for download, together with an abstract and a citation each generated from the metadata (Fig. 2). Proceedings are represented as a whole and by individual contributions.


SPL IV: Extended Fortran IV For Process Control

Autor(en): Oerter, C.W. [DBLP] ; Peterman, G.L. [DBLP]

Zusammenfassung

Extensions to Fortran are presented to illustrate the value of building into the language facilities for known requirements.

Vollständige Referenz | BibTeX

Oerter, C. & Peterman, G., (1970). SPL IV: Extended Fortran IV For Process Control. Fifth Annual Workshop on The Use of Digital Computers in Process Control. Lousiana State Univ.

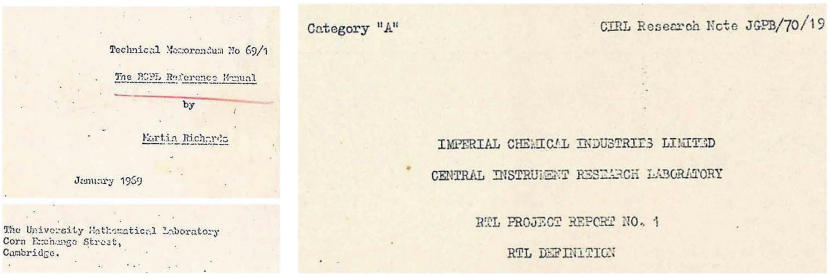
Fig. 2. Abstract and generated citation for SPL IV

5 Content of the Archive

The heterogeneous and somewhat arbitrary stock is artificially arranged here to make it easier to read sequentially, even if the subareas overlap in terms of content. The topics are Basics, Scientific Papers, Events, Project PDV, PEARL Rundschau, and Results. The areas are presented individually. Some particular objects are presented with front matter screen shots in Figs. 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12.

5.1 Basics

Language Descriptions (“Sprachbeschreibungen”). This section contains drafts for real-time and system languages. Historic highlights are BCPL, the predecessor of B and of C, by Cambridge University (1969) [1] (Fig. 3 left), RTL by Imperial Chemical Industries (1970) [4] (Fig. 3 right) and, out of competition, Fortran for Siemens 4004 (1965).


Technical Memorandum No 69/1

The BCPL Reference Manual

by

Martin Richards

January 1969

The University Mathematical Laboratory
Corn Exchange Street,
Cambridge.

Category "A" CIRL Research Note JGPB/70/19

IMPERIAL CHEMICAL INDUSTRIES LIMITED

CENTRAL INSTRUMENT RESEARCH LABORATORY

RTL PROJECT REPORT NO. 1

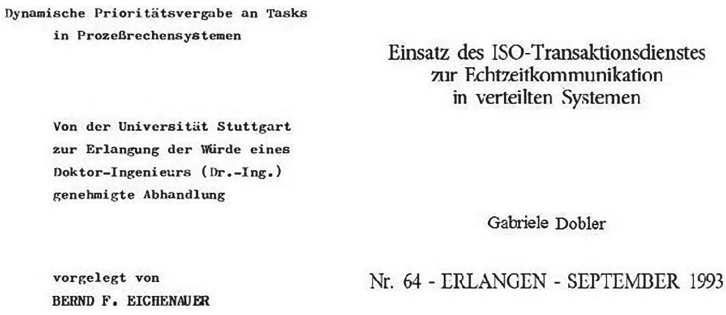
RTL DEFINITION

Fig. 3. Language descriptions, left: BCPL 1969 [1], right: RTL 1970 [4]

PEARL Language Descriptions (“PEARL Sprachbeschreibungen”). Here one can find company-independent PEARL language descriptions from the early days and from the time of redesign.

5.2 Scientific Material

PhD Theses ("Dissertationen"). Start and establishment of the topic real-time systems, from language definition to the development of design and test systems, was not possible without qualified scientific support. At a number of universities, this was reflected by relevant PhD theses, often in the then new discipline of computer science. The documents that can be called up here are not put together representatively, but rather come from random collections. An early example is the thesis on dynamic priority assignment (1975) [7] (Fig. 4 left), a later work (1993) [14] (Fig. 4 right) deals with mapping the ISO transaction service to real-time communication.


Dynamische Prioritätsvergabe an Tasks
in Prozeßrechensystemen

Von der Universität Stuttgart
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

vorgelegt von
BERND F. EICHENAUER

Einsatz des ISO-Transaktionsdienstes
zur Echtzeitkommunikation
in verteilten Systemen

Gabriele Dobler

Nr. 64 - ERLANGEN - SEPTEMBER 1993

Fig. 4. Theses, left: Eichenauer 1975 [7], right: Dobler 1993 [14]

PEARL Papers ("Volltexte"). Here one finds scientifically published articles on trends in PEARL development in the initial phase (1970s) [2] and in the redesign phase (1980s).

5.3 Events

Siemens User Group ("Siemens Anwenderkreis – SAK"). The user groups of German manufacturers of process control computers, e.g. the Siemens user group (SAK), witnessed early real-time applications. The annual user conferences were organised by universities, research institutions or the manufacturers themselves. They were the fora for exchanging information about applications, software and developments. The proceedings of these conferences reflect the pioneering character of the early 1970s and report, for example, on the computer-aided control of the first German research satellite AZUR (1969/1970) [3] (Fig. 5).

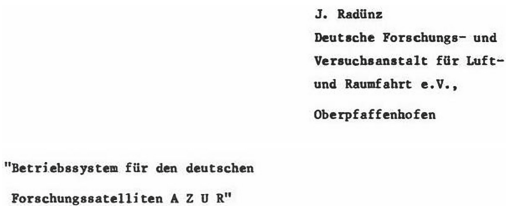


Fig. 5. Control of the AZUR satellite 1970 [3]

PEARL Workshop ("PEARL Tagung"). Since the early 1980s, the PEARL Association (PEARL-Verein, PV) held annual workshops on PEARL in particular and real-time systems in general. This type of series was adopted by GI in 1991. The workshop proceedings were published independently until 1988, and from 1989 on by Springer-Verlag. In 1987, Hewlett Packard published a pioneering article on adding real-time capabilities to Unix [4] (Fig. 6).

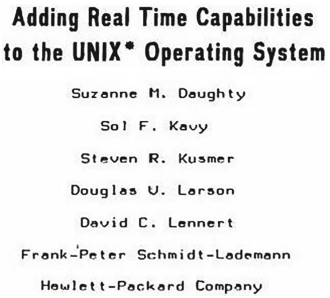


Fig. 6. Real-time Unix by HP 1987 [4]

Personal Real-Time Computing. With the emergence of personal computers in the mid-1980s, traditional process control computers were pushed into the background. The PEARL Association seized this trend with annual workshops on "Personal Real-time Computing". A contribution from 1985 about the use of personal computers to control electrical energy supply seems ahead of its time and somewhat strange.

Proceedings. This section of the archive contains the proceedings of an early international conference on real-time systems held in Germany in 1972 [5] (Fig. 7).

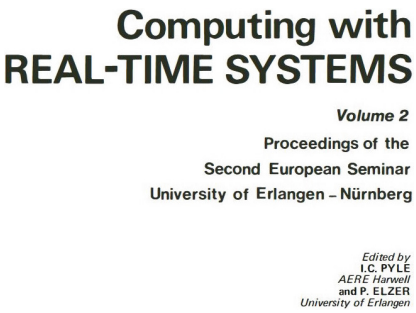


Fig. 7. Proceedings edited by Pyle, Elzer 1972 [5]

5.4 The Project "Process Control by Data Processing" ("Prozesslenkung Durch Datenverarbeitung – PDV")

The project PDV as a part of the Federal Government's data processing programme [6] brought about the greatest development push for real-time systems in Germany. The project's results were published in a series of reports/development notes (Berichte/Entwicklungsnotizen) and communications/project reports (Mitteilungen/Projektberichte) by the organisation managing the project, i.e. Kernforschungszentrum Karlsruhe (KfK). These documents reflect the full breadth of the development process at that time. For example, report 76 describes the virtual assembler CIMIC used in implementations, report 78 a portable real-time operating system for the (8-bit) microprocessor Z80, and report 80 applications such as a cable coating process, leak monitoring in pipelines, and monitoring the wheel load conditions of rail vehicles in rolling operation. Report 132 from 1979 presents future prospects. Here the use of real-time systems in telecommunications is propagated [8] (Fig. 8).

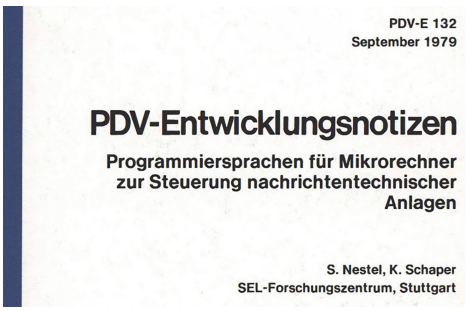
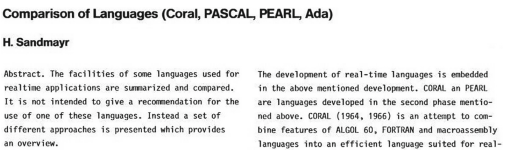


Fig. 8. Real-time systems for telecommunication by SEL 1979 [8]

5.5 PEARL Review ("PEARL-Rundschau")

After the funding of the PEARL development had ceased, the development activities were bundled within the PEARL Association based at the Society of German Engineers (VDI) in Düsseldorf. Among other papers, the Association published the PEARL-Rundschau (1980 to 1982). Over time, the view of real-time systems in general broadened, as can be seen in the article [5] (Fig. 9) with a comparison of the languages Coral, Pascal, PEARL, and Ada in Volume 2, Issue 2 (1981). A good overview of the already wide range of applications can be found in the article entitled Industrial Applications of PEARL (Volume 3, Issue 1, 1982), featuring automation of soaking furnaces, electrical power distribution, data communication, and online coordinate transformation for industrial robots [6] (Fig. 10).



Comparison of Languages (Coral, PASCAL, PEARL, Ada)

H. Sandmayr

Abstract. The facilities of some languages used for realtime applications are summarized and compared. It is not intended to give a recommendation for the use of one of these languages. Instead a set of different approaches is presented which provides an overview.

The development of real-time languages is embedded in the above mentioned development. CORAL an PEARL are languages developed in the second phase mentioned above. CORAL (1964, 1966) is an attempt to combine features of ALGOL 60, FORTRAN and macroassembly languages into an efficient language suited for real-

Fig. 9. Real-time language comparison by Sandmayr 1981 [5]

Industrial Applications of PEARL

Dr. H. Steusloff, Karlsruhe (IITB)

Summary

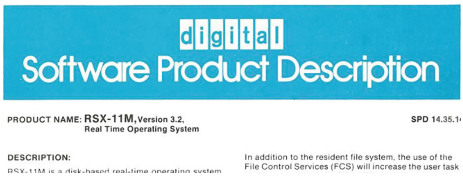
The value of a programming language may only be determined by application experience. The language PEARL (Process and Experiment Automation Realtime Language) [1] has been desig ed to be an application programming language

Fig. 10. Industrial Applications by Steusloff 1982 [6]

5.6 Results

System Software ("Systemsoftware"). This section of the archive contains descriptions of and references to PEARL and other real-time systems, i.e. compilers and operating systems. It is striking to see how consistently the U.S. pioneer Digital Equipment had set up its commitment to PEARL [9] in the 80ies.

Reports ("Berichte"). This section contains detailed final reports of larger real-time projects funded before and after the PDV period.

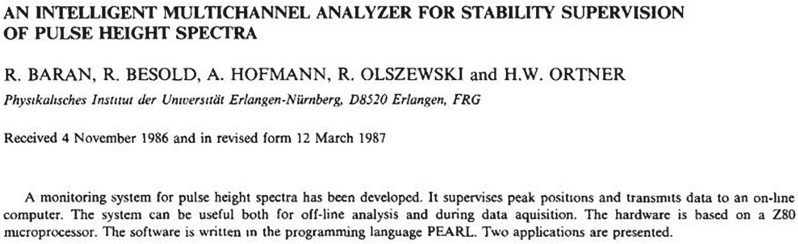

digital
Software Product Description

PRODUCT NAME: RSX-11M, Version 3.2, Real Time Operating System SPD 14.35.1

DESCRIPTION: In addition to the resident file system, the use of the File Control Services (FCS) will increase the user task

Fig. 11. RSX11M operating system by Digital Equipment 1980 [9]

Application Reports ("Anwendungsberichte"). Reports on early real-time applications cannot only be found in relevant proceedings or as journal articles, but also distributed over various non-IT-oriented places. It would be difficult to find them there without detailed knowledge of authors or topic. The collection in the archive should help. It ranges from laboratory reports to regular publications. Reports on deployments in television, in high-energy physics [12] (Fig. 12), or in recording aircraft noise are striking.


AN INTELLIGENT MULTICHANNEL ANALYZER FOR STABILITY SUPERVISION OF PULSE HEIGHT SPECTRA

R. BARAN, R. BESOLD, A. HOFMANN, R. OLSZEWSKI and H.W. ORTNER

Physikalisches Institut der Universität Erlangen-Nürnberg, D8520 Erlangen, FRG

Received 4 November 1986 and in revised form 12 March 1987

A monitoring system for pulse height spectra has been developed. It supervises peak positions and transmits data to an on-line computer. The system can be useful both for off-line analysis and during data aquisition. The hardware is based on a Z80 microprocessor. The software is written in the programming language PEARL. Two applications are presented.

Fig. 12. A multi-channel analyser in high-energy physics 1987 [12]

6 Usage

The real-time systems archive can be reached by invoking https://dl.gi.de and, then, descending to departments ("Fachbereiche") → technical informatics ("Technische Informatik") → real-time systems ("Echtzeitsysteme").

Searching by means of listing ("Auflistung") leads either to title, author, publication date or keyword. A search mask can be used to search locally or globally. A local search, e.g. for "Application", returns all articles with the searched term in the title or the text. Page layout and sorting are adjustable.

7 Experience

The real-time archive is found by internet search engines with high priority. If you search for a rather unusual term like "Project PDV", the engines will

generally return the searched result collected from the archive on the first page, although the term PDV is now used elsewhere.

Unfortunately, due to its origins, the archive is steeped in the past. Newer contributions – or even historical ones – are very welcome, especially relevant PhD theses. If you have any, please contact `archiv@real-time.de`.

References

1. Baran, R., Besold, R., Hofmann, A., Olszewski, R., Ortner, H.W.: An intelligent multichannel analyzer for stability supervision of pulse height spectra. Nucl. Instrum. Methods Phys. Res. **258**(1), 91–94 (1987)
2. Brandes, J., et al.: The concept of a process-and experiment-oriented programming language. elektronische datenverarbeitung **10**, 429–442 (1970)
3. Daugthy, S.M., Kavy, S.F., Kusmer, S.R., Larson, D.V., Lennert, D.C., Schmidt-Lademann, F.-P.: Adding real time capabilities to the UNIX* operating system. In: Proceedings of the PEARL 1987, pp. 151–165. PEARL-Verein/GMA, Düsseldorf (1987)
4. Dobler, G.: Einsatz des ISO-Transaktionsdienstes zur Echtzeitkommunikation in verteilten Systemen. Dissertation, University of Erlangen-Nürnberg, Erlangen (1993)
5. Eichenauer, B.: Dynamische Prioritätsvergabe an Tasks in Prozeßrechensystemen. Dissertation, University of Stuttgart, Stuttgart (1975)
6. Nestel, S., Schaper, K.: Programmiersprachen für Mikrorechner zur Steuerung nachrichtentechnischer Anlagen. PDV-Entwicklungsnotizen PDV-E 132. KfK, Karlsruhe (1979)
7. NN: Forschungsbericht (IV) der Bundesregierung. Bundesminister für Bildung und Wissenschaft, Bonn (1972)
8. NN: RSX-11 M Real Time Operating System. Digital Equipment Corporation, Maynard (1980)
9. NN: RTL Definition. Project report, Imperial Chemical Industries Ltd., Reading (1970)
10. Pyle, I.C., Elzer, P.: Computing with Real-Time Systems, vol. 2. Transcripta Books, London (1972)
11. Radünz, J.: Betriebssystem für den deutschen Forschungssatelliten AZUR. In: Proceedings of Siemens-Prozeßrechner-Benutzertagung 1970, pp. 76–102. KFA Jülich, Jülich (1970)
12. Richards, M.: The BCPL reference manual. Technical Memorandum. The University Mathematical Laboratory, Cambridge (1969)
13. Sandmayr, H.: Comparison of Languages (Coral, PASCAL, PEARL, Ada). PEARL-Rundschau Band 2 Nr. 2, pp. 29–36. PEARL-Verein/GMA, Düsseldorf (1981)
14. Steusloff, H.: Industrial Applications of PEARL. PEARL-Rundschau Band 3 Nr. 1, pp. 35–41. PEARL-Verein/GMA, Düsseldorf (1982)

Deadlock Detection in OpenPEARL

Jan Knoblauch(✉)

Hochschule Furtwangen University, 79761 Waldshut-Tiengen, Germany
jan.knoblauch@gmx.de

Abstract. OpenPEARL is an open source build system for PEARL, a DIN-standard programming language designed for building multitasking and real-time applications. In the target domain of PEARL applications, error-free synchronization of multiple processes is of particular importance. Among other things, deadlocks pose a great risk, as they usually occur irregularly and cause the system to crash. The following describes a concept that can detect deadlocks that may occur and those that have already occurred through various approaches. It is integrated with OpenPEARL and is designed to help application developers identify, understand, and troubleshoot process synchronization errors.

Keywords: PEARL · OpenPEARL · deadlock · static analysis · dynamic analysis · control flow graph · resource allocation graph

1 Deadlock

Deadlocks represent a danger in application development that should not be underestimated. They occur when each process of a set of processes waits for an action of another of these processes. Often these actions are the release of resources such as files, synchronization locks, interfaces or peripherals. Deadlocks are usually dependent on several factors that are not under the control of the application. Due to external factors such as non-deterministic scheduling between processes and different latencies in network and hardware accesses, deadlocks can occur very irregularly. Deadlocks are usually not detectable by common testing methods such as unit and module tests. The occurrence of deadlocks may differ in development, test and production environments, for example, if factors such as system performance and load are different. If a deadlock occurs regularly, it is probably discovered and fixed during development or testing. But if the occurrence of a deadlock depends on several rather unlikely input data, function calls, external applications or services, the problems may not be detected.

In a deadlock situation, all involved processes are not executed any further, and a deadlock exists that normally cannot be resolved in an automated way. If a deadlock thus occurs, the system can usually not continue to work, but must be restarted. External intervention is needed, whether manual or from a monitoring system. Deadlocks are therefore considered critical faults that are difficult to find through testing, and often lead to failures when they do occur. Especially with high reliability requirements, it must therefore be possible to systematically exclude deadlocks.

For a deadlock to occur, [2] states that four conditions are sufficient.

H. Unger and M. Schaible (Eds.): Real-Time 2022, LNNS 674, pp. 13–22, 2023.
https://doi.org/10.1007/978-3-031-32700-1_2

1. No Preemption: Resources are released exclusively by the processes themselves
2. Hold and Wait: The processes already have resources and still need more
3. Mutual Exclusion: Only one process can access a resource at the same time
4. Circular Wait: A circular dependency exists between at least two resources of two processes

For a deadlock to occur, it is sufficient if these four sufficient conditions are met. The conditions imply a deadlock, but there is no equivalence between these conditions and the occurrence of a deadlock. Therefore, deadlocks can also occur if not all of these conditions are met.

Several ways exist to avoid deadlocks during application development. However, it depends on the developers whether and how carefully these mechanisms are implemented. In [1], deadlock treatment strategies are classified into the three following categories.

1.1 Deadlock Prevention

The deadlock prevention approach describes the prevention of necessary conditions for deadlocks on a logical level. According to [4], it is sufficient to enforce one of the following approaches.

The condition "Hold and Wait" is not satisfied if all resources must be requested simultaneously, which is described by the approach "Simultaneous Locking". A process must request all resources it may need at the same time and cannot request resources again until it has released all occupied resources. However, this severely limits parallelism when a process requires one resource for only a short period of time and another resource for a comparatively long period of time. According to the principle of simultaneous locking, the process would have to occupy both resources at the same time, even if it does not use the resource required for a short time until much later. During this time, this resource would also not be available to the other processes and would therefore remain unused. This prevents the optimal allocation of resources and therefore severely limits parallelism.

In contrast, the approach "Ordered Locking" ensures that all resources can only be requested in a specific order. For this, a sequence must be formed over all resources of a system. A process can only request resources that have a higher order in this sequence than the resources that the process already occupies. In order to occupy a lower-order resource, the process must first release its occupied higher-order resources. This means that no cyclical dependencies can arise between processes and their operating resources. Thus, the condition "Circular Wait" is no longer fulfilled.

1.2 Deadlock Avoidance

Deadlocks can also be actively avoided during the execution of an application, which is called "Deadlock Avoidance". Thereby, the scheduling of the processes

is controlled in such a way that there is always a sequence in which all upcoming resource requests can be executed.

A method to avoid deadlocks was introduced by Edsger W. Dijkstra in 1965 as "Bankers Algorithm" and is described in [4]. Adapted from a banker who grants loans of his customers, it controls the allocation of resources. At the beginning each process must be known, and which resources it will need during the execution. The algorithm describes that before each resource allocation, it is checked whether the application is still in a safe state afterwards. If this is the case, the resource is allocated, otherwise the allocation is delayed out, even if it would be feasible at the current time. A state is considered safe if a scheduling order is possible in which all requested resources can be allocated without jams. If this is not the case, the state is unsafe. Depending on the scheduling, allocations and releases of the processes, deadlocks can occur in an unsafe state.

By executing or delaying possible resource allocations through the deadlock avoidance depending on the state, the use of it actively changes the behavior of resource allocations and thus the runtime behavior of the entire application. The PEARL definition is to allocate requested resources when they are available. However, by delaying possible resource allocations through deadlock avoidance, this would affect the runtime behavior of the application. This approach will therefore not be examined further.

1.3 Deadlock Detection and Resolution

The third category of deadlock treatment strategies is the detection of deadlock situations that have occurred during execution, as well as their resolution. To resolve a deadlock, a resource must be withdrawn from a participating task. Basically, the resource can only be revoked if the operations that the task has applied to the resource are undone. In order to be able to revoke a resource from an PEARL task according to this principle, the task's operations on the resource must be able to be undone. In this context, the synchronization mechanisms of PEARL do not represent the actual resources, but in the field of embedded systems, for example, these resources are motors and actuators. The synchronization mechanisms merely serve to control which task may use which resource. Even if the generated actions of a task on an actuator were logged and undone in the event of a problem, this would not be possible in a generally valid manner, and certainly would not make sense. For this reason, it is generally not possible to withdraw a resource from a task, and therefore deadlocks that have occurred cannot be resolved automatically in PEARL (and other general-purpose languages) even at the logical level.

2 OpenPEARL

The standardized[1] programming language PEARL ("Process and Experiment Automation Realtime Language") provides convenient support for requirements

[1] DIN 66253:2018-03.