بررسی یک معماری سیستم عامل برای سیستمهای توکار[1] خودرو

سعید جلیلی
استادیار گروه مهندسی کامپیوتر
دانشکدهٔ فنی و مهندسی، دانشگاه تربیت مدرس
تلفن:(۳۳۷۴) ۸۰۱۱۰۰۱
E-mail: sajalili@modares.ac.ir

محمد امین علی پور
دانشجوی کارشناسی ارشد مهندسی کامپیوتر
دانشکدهٔ فنی و مهندسی، دانشگاه تربیت مدرس
تلفن:(۳۵۲۱)۸۰۱۱۰۰۱
E-mail: alipourm@modares.ac.ir

چکیده:حجم کم، توان محاسباتی بالا و قیمت مناسب ریزپردازنده ها، محققان و صنعتگران را بر آن داشته است از ریزپردازنده ها در کاربردهایی غیر از کاربردهای متداول در سیستمهای کامپیوتری استفاده کنند. در این کاربردها ریزپردازنده درون یک سیستم مکانیکی قرارگرفته و مدیریت جزء یا اجزایی از سیستم را بر عهده می گیرد. به این گونه سیستمها، سیستمهای توکار می گویند.

عواملی که باعث گرایش مهندسان به این سیستمها شده است عبارتند از: (۱) انعطاف پذیری این سیستمها بواسطهٔ قابلیت برنامه ریزی ریز پردازنده ها ،(۲) کاهش پیچیدگی سیستم به خاطر خلاصه شدن پیچیدگی سیستم در یک الگوریتم کد شده در ریز پردازنده که نهایتاً منجر به آزمون پذیری و اصلاح آسانتر سیستم می گردد. از سیستمهای صوتی و تصویری و مایکروفرهای قابل برنامه ریزی گرفته تا سیستمهای هدایت موشک، همگی مثالهایی از سیستم توکار هستند.

خودرو نیز از این قاعده مستثنی نبوده است. لیموزین[2] امروزی بالغ بر بیست ریزپردازنده در خود جای داده است، که هر یک وظیفه ای خاص بر عهده دارند مانند: کنترل ترمزها، کنترل سیستم تهویهٔ هوا و... . استفاده از تکنولوژی ریزپردازنده ها و سیستمهای توکار به شدت در حال فزونی است. تا آنجا که گفته می شود "خودروهای آینده روی ریزپردازنده ها حرکت می کنند."

در سیستمهای توکار حافظهٔ محدود ، مهمترین چالش طراحان است. طراحان این سیستمها بایستی الگوریتمهای خود را با توجه به فضای محدود حافظهٔ ریزپردازنده طراحی و پیاده سازی کنند. در این مقاله یک معماری سیستم عامل برای سیستمهای توکار خودروها ارائه شده و مورد بررسی و ارزیابی قرار می گیرد.

۱-مقدمه

حجم کم، توان محاسباتی بالا و قیمت مناسب ریزپردازنده ها، محققان و صنعتگران را بر آن داشته است از ریزپردازنده ها در کاربردهایی غیر از کاربردهای متداول در سیستمهای کامپیوتری استفاده کنند. در این کاربردها

[1] Embedded Systems
[2] limousine

ریزپردازنده درون یک سیستم مکانیکی قرارگرفته و مدیریت جزء یا اجزایی از سیستم را بر عهده می گیرد.به عبارت دقیقتر سیستمهای توکار ترکیبی از نرم افزارها و سخت افزارهای محاسباتی با انواع دیگر اجزاء می باشند تا هدف معینی را محقق سازند. از سیستمهای صوتی و تصویری و مایکروفرهای قابل برنامه ریزی گرفته تا سیستمهای هدایت موشک، همگی مثالهایی از سیستم توکار هستند.

عواملی که باعث گرایش مهندسان به این سیستمها شده است عبارتند از: (۱) انعطاف پذیری این سیستمها بواسطهٔ قابلیت برنامه ریزی ریز پردازنده ها ،(۲) کاهش پیچیدگی سیستم به خاطر خلاصه شدن پیچیدگی سیستم در یک الگوریتم کد شده در ریز پردازنده که نهایتاً منجر به آزمون پذیری و اصلاح آسانتر سیستم می گردد. صنایعی مانند هوانوردی و ماشین سازی نیز از رسوخ این سیستمها در امان نبوده اند. لیموزینهای[1] امروزی را می توان یک سیستم بزرگ توزیع شده روی چرخ دانست. خودروهایی وجود دارند که در کارهای خود از بیش از ۲۰ ریزپردازندهٔ (یا میکروکنترلر) متصل به هم استفاده می کنند.

هرچند که این سیستمها با توجه به تعداد ریزپردازنده ها بسیاز بزرگ هستند، اما سیستم از نظر حجم کل حافظه بسیار در مضیغه است. مثلاً در نمونهٔ بیان شده با ۲۰ میکروکنترلر، حجم کل حافظهٔ سیستم در حدود ۱-۲ مگابایت می باشد. این مساله چیز عجیبی نیست زیرا یک واحد کنترلی الکترونیکی[2] تنها ۴ کیلوبایت حافظه دارد. برنامه کاربردی این سیستم نیز به تنهایی نمی تواند پیچیدگی این معماری توزیع شده را اداره کند.بنابراین سیستم عامل توکار ویژه ای مورد نیاز است تا مدیریت صحیح، انعطاف پذیری و حتی کارآیی سیستم را تضمین کند. و در همین حال با توجه به محدودیت حافظه، سیستم عامل و برنامهٔ کاربردی که باید روی آن اجرا شود نباید از چند کیلو بایت تجاوز کنند.

یک نمونه از این سیستمها توسط استانداردOSEK/VDX [۱۴] بیان شده است. این استاندارد ویژگیهای یک سیستم عامل توکار را برای صنایع خودروسازی بیان می کند. با در نظرگیری این استاندارد صنایع اروپایی می کوشند تا نرم افزار های مورد نیاز سیستمهای توکار خود را بسازند. متاسفانه یک استاندارد به تنهایی نمی تواند تمام نیازهای صنایع خودرو سازی را (بدون کاستن از کارآیی یا کار[3]) برآورده سازد. به همین دلیل OSEK چهار کلاس تطابق[4] تعریف کرده است که هر یک مجموعه ای از فعالیتهای سیستم عامل را ارائه می دهند. طراح سیستم با توجه به مشخصات مورد نظر خود یکی از این کلاسها را به عنوان استاندارد مشخص می کند.

PURE معماری دیگری است که نه تنها سعی در وفق دادن خود با محیطهای توکار دارد؛ بلکه سعی می کند خود را با خود را با کابرد مورد نظر تنظیم کند. هدف PURE آن است که یک سیستم به شدت قابل تنظیم[5] را بسازد و به طراح سیستم اجازه دهد تا قابلیتهای عملیاتی مورد نظر خود را، خود انتخاب کند. یعنی در ضمن اینکه PURE به هیچ کاربردی محدود نیست، این امکان را فراهم می کند که کاربردهای مورد نیازمحیطهای توکاری که با مشکل محدودیت منابع(حافظه) دارند؛ به خوبی اجرا شوند.

---

[1] Limousine
[2] Electronic Control Unit
[3] Functionality
[4] Conformance Class
[5] Highly configurable

در بخش ۲ کارهای مرتبط مورد بررسی قرار گرفته ، در بخش ۳ معماری PURE توضیح داده خواهد شد و در پایان نیز برخی نتایج عملی آن بیان شده و مورد بررسی قرار می گیرد.

۲- کارهای مرتبط

از آنجا که استراتژیهای پیاده سازی شده در سیستم عاملهای متداول برای کاربرد خاصی بهینه نشده اند؛ عموما کارآیی کاربر را محدود می سازند. به همین دلیل در کاربردهایی مانند سیستمهای بی درنگ[1] و سیستمهای توکار از سیستم عاملهای خاص استفاده می شود. در واقع این گونه سیستم ها به بسیاری از توابعی که در این سیستم عامل های متداول پیاده سازی شده اند نیاز ندارند و انتظار دارند که بخش اندک مورد استفادهٔ آنها بصورت بهینه پیاده سازی شده باشد.

سیستمهای عامل قابل توسعه[2] به منظور حل این مساله به طراح سیستم اجازه می دهند تا توابع مورد نیاز خود رابه هستهٔ سیستم عامل[3] افزوده و یا توابع آنرا تغییر دهند. مثلاً در SPIN [۲] توابع سطح کاربرد می توانند به هسته اضافه شوند.در مقابل ایدهٔ Exokernel [۷] آن است که تا آنجا که امکان دارد تابع در اختیار لایهٔ کاربرد قرار گیرد تا طراح بتواند یکی از آنها را انتخاب کند. VINO به کمک سازماندهی هستهٔ سیستم عامل به صورت مجموعه ای از انتزاعها با قابلیت مجدد و قابل توسعه سعی در استفادهٔ مجدد کد دارد.

هر چند که سیستمهای فوق تلاش می کنند که خود را در جهت تامین نیاز برنامهٔ کاربردی تنظیم کنند، ولی در محیطهای توکار کارآیی ندارند. و یا به عبارت بهتر هدف آنها اجرا در محیطهای توکار نیست. در این نوع محیطها (توکار) نرم افزار باید در قالب محدودیتهای شدید مانند حجم RAMو یا ROM کار کند. در این محیطها اجرای همزمان چند برنامه بر روی یک میکروکنترلر کمتر مورد توجه است زیرا معمولاً میکروکنترلرها در حالت تک کاربره کار می کنند.

محققان دیگری بر روی فراهم آوردن مجموعه های ساخت سیستم عامل کار می کنند. در این رویکرد سعی بر آن است که اجزای مختلف سیستم عامل بطور جداگانه و در پیاده سازی های متنوع در اختیار طراح قرار گیرد تا او به اختیار خود سیستم عامل مورد نظر خود را بسازد. PURE و OSKit [۹] نمونه هایی از این رویکرد هستند. OSKit با در اختیار گذاشتن تعداد زیادی از اجزا مختلف امکان اسمبل کردن و ساخت یک سیستم عامل کامل را فراهم می آورد. در OSKit این اجزا باید از هم مجزا باشند و از طریق برقراری یک نظام مطلوب واکشی[4] با هم ارتباط داشته باشند. در مقابل PURE از مفهوم مجموعهٔ ساخت سیستم تنها در سطح طراحی و تنظیم استفاده می کند. یعنی در پیاده سازی به خاطر محدویت شدید منابع سیستم، هیچ یک از مرزهای مصنوعی انتزاع ها رعایت نمی شود. و یا به عبارت بهتر "فقط طراحی سلسه مراتبی است نه پیاده سازی." رویکرد PURE برخورد با سیستم عامل به صورت "خانواده ای از برنامه ها" و پیاده سازی آنها به روش شیءگرایی می باشد.

---

[1] Real-time Systems
[2] Extensible Operating Systems
[3] Operating System Kernel
[4] Well-defined invocation mechanism

Trigger خانواده ای از "سیستمهای پشتیبانی شیء[1]"است[۳] که دستهٔ زیادی از کاربردها را از جمله سیستم توکار هدف گرفته است. فرق اصلی آن با PURE آن است که بر روی اشیای توزیع شده و مانا[2] تمرکز دارد، حال آنکه PURE بر تنظیمات ظریف هستهٔ سیستم عامل با در نظر گرفتن محدودطتهای موجود تاکید دارد.
ERCOS یک سیستم عامل شیءگرا برای کاربردهای خودرو می باشد[۱]. که به کمک سازماندهی کاربرد به صورت تعداد زیادی شیء از پیش تعریف شده و ارتباط آنها از طریق پیامهای وضعیت، طراحی واضح محقق شده است. به منظور تامین محدودیتهای سیستم، بهینه سازی کد بصورت off-line توسط ابزارهای مختلف صورت گرفته است.
سیستمهای دیگر تجاری با بهره گیری از ساختار ماژولار، به سازندهٔ برنامهٔ کاربردی اجازه می دهد که با توجه به نیازهای خود، سیستم را تنظیم کند. برای مثال VxWorks [۸]، pSOS [۱۱] و QNX [۱۶] همگی یک هستهٔ چندوظیفه ای بی درنگ را تهیه می کنند که بر روی آنها می توانند اجزای مستقل قرار گیرند. هر چند که این سیستمها سعی در کم کردن هسته داشته اند، اما خود هسته قابل تنظیم نیست. درکل این سطستمها بسطار بزرگتر از آنند که بتوانند در کاربردهای صنایع اتومبیل سازی به کار روند.

۳- معماری سیستم عامل PURE

PURE به جای معرفی یک معماری جدید، به طراح اجازه می دهد که انواع مختلف معماری را بسازد.به همین خاطر معماری ویژه ای در طراحی معرفی نمی شود. بلکه مجموعه ای از عناصر ساخت سیستم عامل معرفی می شود. تصمیم اینکه سیستم عاملMonotonic باشد یا Micro Kernel بر عهدهٔ طراحی است که از اجزای PURE جهت ساخت سیستم استفاده می کند.PURE یک سیستم عامل باز است. تمام انتراعهای آن برای طراح و برنامه نویس روشن است.

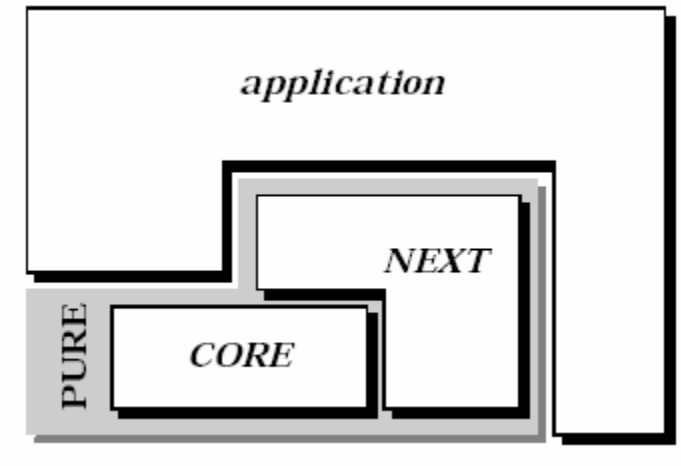
به منظور تامین نیازهای برنامه و کاهش حجم حافظهٔ مورد استفاده، PURE به صورت "خانواده ای از برنامه ها[3]" طراحی شده است. هدف آن است که تمام اجزاء مورد نظر انتخاب و بقیه حذف شوند. این امر با تهیهٔ یک ساختار ریز–دانه[4] و با قابلیت تغییر بسیار تحقق یافته است. بر اساس مفهوم "زیر مجموعهٔ مینیمال[5]" از توابع سیستم هستهٔ اولیهٔ سیستم ساخته شده و با "توسعه های مینیمال[6]" سیستم بطور افزایشی گسترش می یابد. در هر گام یک پایهٔ مینیمال[7] برای توسعهٔ سطوح بالاتر ساخته می شود.
از آنجا که توسعه ها تنها بر اساس نیاز ساخته می شود و تصمیمهای طراحی تا آنجا که ممکن است به تاخیر می افتد، یک سیستم کاملا مبتنی بر کاربرد است. و نتیجهٔ این خواهد بود که کاربرد آخرین توسعهٔ سیستم خواهد

[1] Object-support system
[2] Persistent
[3] Program family
[4] Fine-Granular
[5] Minimal Subset
[6] Minimal extension
[7] Minimal Basis

بود. بدین ترتیب مرز قدیمی بین کاربرد و سیستم عامل محو می شود. سیستم عامل به کاربرد توسعه می یابد. بنابراین دیگر لازم نیست برای منابعی که استفاده نمی کند بهایی بپردازد.

PURE با استفاده از با استفاده از کلاسهای C++ پیاده سازی شده است. هر کلاس یک انتزاع داده ای را می سازد.از وراثت برای ساخت انتزاعهای پیچیده استفاده شده است. به خاطر مفهوم خانواده، توسعه سیستم در گامهای بسیار کوچک ساخته می شوند.برای مثال، بلاک کنترل نخها[1] از ۴۵ کلاس که در ۱۴ سطح سلسله مراتب قرار گرفته اند، ساخته شده است.

در شکل ۱ ساختار کلی PURE آمده است. PURE متشکل از هسته و توسعه های هسته می باشد. هسته جهت اجرای همروند شیء های فعال و غیر فعال،CORE را پیاده سازی می کند. به کمک توسعه های مینیمال،NEXT ، موارد مانند مدل پردازه و فضای آدرس دهی مربوط به کاربرد، همگام سازی نخها و ارسال پیام را پیاده سازی می کند. از این توسعه ها تنها در صورتی استفاده می شود که برنامهٔ به آن نیاز داشته باشد.



شکل ۱- ساختار کلیPURE

۳-۱- هسته

COREزیر مجموعهٔ مینیمال توابع سیستم را برای زمانبندی رخدادها و فعالیتها پیاده سازی می کند. رخدادها بصورت وقفه[2]و فعالیتها بوسیلهٔ نخ مدل شده اند.همانطور که در شکل ۲ دیده می شود این توابع در قالب چهار بلاک زیر تعبیه شده اند:

۱- Sluice ، انتزاعی برای همزمانی فعالیتهای مبتنی بر وقفه است.

۲- Flange، انتزاعی برای انتساب شیء ها به بردار وقفه و انتشار رخدادهای غیرمترقبه به سطوح بالاتر است.

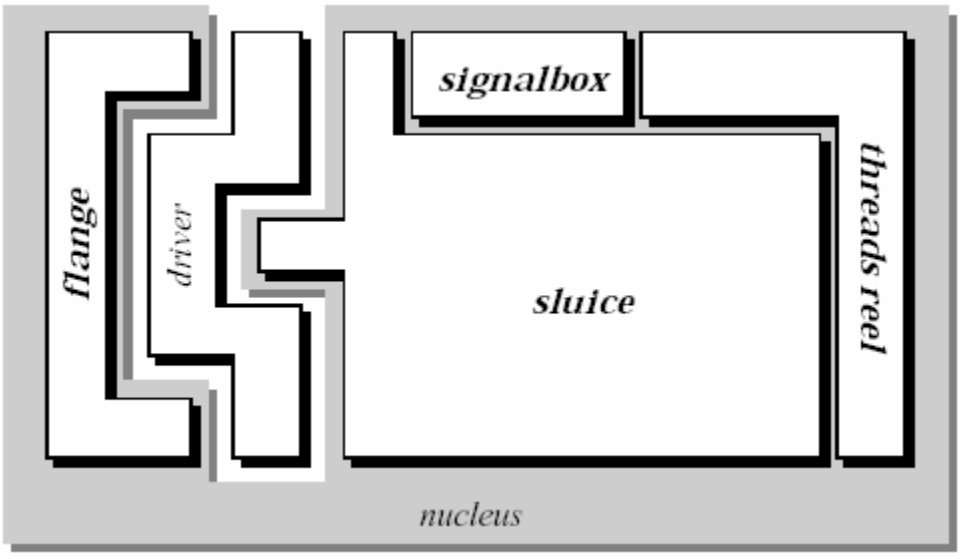
۳- Signal-box، انتزاعی برای همگام سازی شیء های فعال(نخها) است.

---

[1] Thread
[2] Interrupt

۴- Thread-reel، انتزاعی است برای ساخت، زمانبندی و تخریب نخهای فعال است.

در شکل ۲ بلاک پنجمی نیز وجود دارد، این بلاک driver است که باید در طراحی کل سیستم مدنظر قرار گیرد. این بلاک که توابع مربوط به راه اندازهای سیستم را در بر دارد جزئی از هسته نیست بلکه بسته به محیط پیاده سازی در کنار هسته قرار می گیرد.



شکل ۲- ساختار CORE

۳-۲-توسعهٔ OSEK

OSEK یک استاندارد سیستم عامل برای صنایع خودروسازی است.که توسط سازندگان اروپایی خودرو تعیین شده است. همانطور که قبلاً گفته شد دارای چهار کلاس مختلف استاندارد است که طراح به تناسب سیستم حود می تواند یکی را انتخاب کرده و بر اساس آن سیستم حود را طراحی کند. هر یک از این کلاسهامربوط به نیازها و ویژگیهای خاصی از سیستم های کنترل می باشد.هر کلاس یک مدل پردازه،تخصیص اولویت و الگوی فعالیت پردازه را تعریف می کند. بنابراین OSEK را می توان خانواده از سیستم عامل با چهار عضو به حساب آورد.

PURE این استاندارد را بصورت یک زیر خانواده[1]به نام $PURE_{OSEK}$ پیاده سازی کرده که شامل ۶۱ کلاس C++ می باشد.

۴- تحلیل

سیستم عامل PURE در C++ پیاده سازی شده است و متشکل از ۱۰۰ کلاس است که مجموعاً ۶۰۰ تابع راپیاده سازی کرده است. جدول ۱ میزان مصرف حافظه را در مدهای مختلف پردارشی بیان کرده است. همانطور که دیده می شود PURE به خاطر معماری خاصش حجم بسیار کمی از حافظه را اشغال می کند.

[1] Subfamily

Family member	Size (in bytes)			
	Text	Data	BSS	Total
Interruptedly	812	64	392	1268
Reconcile	1882	8	416	2306
Exclusive	433	0	0	434
Cooperative	1620	0	28	1648
Non-preemptive	1620	0	28	1699
preemptive	3642	8	428	4062

جدول ۱- حافظهٔ مصرفی PURE

۵- نتیجه گیری

محدودیت منابع در محیطهای توکار و نیاز برنامه های کاربردی به یک سیستم عامل به منظور مدیریت درست و کارآی سیستم باعث شده است تا تلاشهایی جهت طراحی و پیاده سازی سیستم های عامل ویژهٔ این نوع محیطها صورت گیرد. در این مقاله به مواردی در این زمینه چون VINO, OSKit, SPIN, Exokernel , Trigger, ERCOS, VxWorks, pSOS, QNX اشاره شد. تجربهای که بطور دقیقتر مورد بررسی قرار گرفت ، PURE بود. PURE با استفاده از مفهوم "خانوادهٔ برنامه" و" توسعه های مینیمال" توسعه یافته است. این سیستم تنها در طراحی از سلسله مراتب استفاده می کند و نه در پیاده سازی. بطوریکه برنامهٔ کاربردی آخرین توسعهٔ سیستم است. $PURE_{OSEK}$ سیستمی است که بر اساس استاندارد سیستم های عامل صنایع خودرو سازی پیاده ساری شده است. در تحلیل این سیستم دیدیم که می تواند ویژگیهای مورد نظر سیستم عامل را در محیطهای توکار با اشغال حجم کمی از حافظه(که بزرگترین گلوگاه سیستمهای توکار است) بر آورده سازد.

منابع

[1] *ERCOS White Paper*. Schwieberdingen, 1997.http://www.etas.de/.

[2] B. N. Bershad, S. Savage, P. Pardyak, E. G. Sirer, M. E. Fiuczynski, D. Becker, C. Chambers, and S. Eggers. Extensibility,Safety and Performance in the *SPIN* Operating System. In *Proceedings of the Fifteenth ACM Symposium on Operating System Principles*, pages 267–284, Copper Mountain Resort, Colorado, 1995.

[3] V. Cahill, C. Hogan, A. Judge, D. O'Grady, B. Tangney, and P. Taylor. Extensible systems - the Tigger approach. In *Proceedings of the SIGOPS European Workshop*, pages 151–153. ACM SIGOPS, Sept. 1994. Also technical report TCD-CS-94-07, Dept. of Computer Science, Trinity College Dublin.

[4] R. Campbell, G. Johnston, and V. Russo. Choices (Class Hierarchical Open Interface for Custom Embedded Systems).*Operating Systems Review*, 21(3):9–17, 1987.

[5] R. H. Campbell and S.-M. Tan. _Choices: An Object-Oriented Multimedia Operating System. In *Proceedings*

of the Fifth Workshop on Hot Topics in Operating Systems (HOTOS V), Orcas Island, Washington, May 1995. IEEE Computer Society.
[6] J. Cordsen and W. Schr¨oder-Preikschat. Object-Oriented Operating System Design and the Revival of Program Families. In *Proceedings of the Second International Workshop on Object Orientation in Operating Systems (I-WOOOS '91)*, pages 24–28, Palo Alto, CA, October 17–18, 1991.
[7] D. R. Engler, M. F. Kaashoek, and J. O'Toole, Jr. Exokernel:An Operating System Architecture for Application-Level Resource Management. In *Proceedings of the Fifteenth ACM Symposium on Operating System Principles*, pages 251–266, Copper Mountain Resort, Colorado, 1995.
[8] J. Fogelin. The VxWorks real-time kernel. Technical report, WindRiver Systems.
[9] B. Ford, G. Back, G. Benson, J. Lepreau, A. Lin, and O. Shivers. The Flux OSKit: A Substrate for Kernel and Language Research. In *Proceedings of the Sixteenth ACM Symposium on Operating System Principles*, pages 38–51, Saint-Malo, France, 1997.
[10] A. N. Habermann, L. Flon, and L. Cooprider. Modularization and Hierarchy in a Family of Operating Systems. *Communications of the ACM*, 19(5):266–272, 1976.
[11] Integrated Systems, Sunnyvale, CA. *pSOSystem System Concepts*, 11 1996.
[13] J. Nolte and W. Schr¨oder-Preikschat. Dual Objects — An Object Model for Distributed System Programming. In *Proceeding of the Eighth ACM SIGOPS European Workshop,Support for Composing Distributed Applications*, 1998.
[14] OSEK/VDX Steering Committe. OSEK/VDX Operating System, Oct. 1997. Version 2.0 revision 1.
[15] D. L. Parnas. Designing Software for Ease of Extension and Contraction. *Transaction on Software Engineering*, SE-5(2), 1979.
[16] QNX Software Systems Ltd. *QNX System Architecture*, 1997. http://www.qnx.com/.
[17] F. Sch¨on, W. Schr¨oder-Preikschat, O. Spinczyk, and U. Spinczyk. Design Rationale of the PURE Object-Oriented Embedded Operating System. 1998. Accepted for the *International IFIP WG 10.3/WG 10.5 Workshop on Distributed and Parallel Embedded Systems (DIPES '98)*.
[19] C. Small and M. Seltzer. Structuring the kernel as a toolkit of extensible, reusable components. In *Proceedings of the 1995International Workshop on Object-Orientation in Operating Systems (IWOOOS '95)*, 1995.