«Некоторые практические аспекты создания надежных программных систем»

Александр Дубинин
Группа компаний YADRO

Особенности программных систем

FUNCTIONALITY

Выполнение функций согласно спецификации (требованиям)

COST

Стоимость, как при разработке, так и при эксплуатации

USABILITY

Удобство использования системы

Снижает риск человеческих ошибок (как при прямом использовании, так и при проектировании других систем)

PERFORMANCE

Производительность

DEPENDABILITY

Надежность

Для программных систем также можно определить:

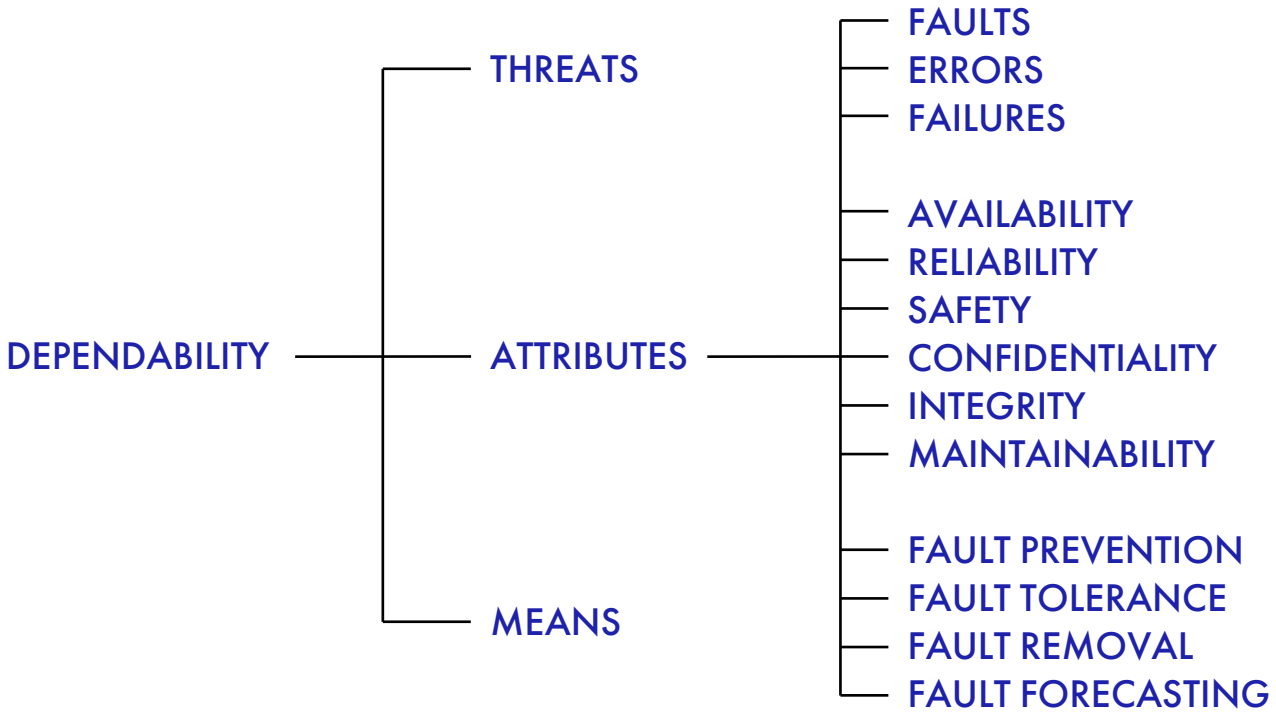
CORRECTNESS **КОРРЕКТНОСТЬ** — Способность системы функционировать согласно спецификации, при условии использования в ее рамках

ROBUSTNESS **ЖИВУЧЕСТЬ** — Способность системы не приносить вреда в случае ошибочного выхода условий работы за рамки спецификации

SECURITY **БЕЗОПАСНОСТЬ** — Способность системы не приносить вреда в случае злонамеренного выхода условий работы за рамки спецификации

Основные понятия надежности

Надежность компьютерной системы — способность выполнять функцию (обеспечивать сервис — т.е. видимое пользователем поведение), результатам которого можно в разумных пределах доверять.



Основные понятия надежности

AVAILABILITY **ДОСТУПНОСТЬ**	Готовность предоставлять сервис, выполнять функцию
RELIABILITY **БЕЗОТКАЗНОСТЬ**	Надежность в смысле достоверности результатов работы, способность давать корректный результат в течении длительного времени
SAFETY **БЕЗОПАСНОСТЬ**	Отсутствие катастрофических последствий для пользователя или окружения
CONFIDENTIALITY **КОНФИДЕНЦИАЛЬНОСТЬ**	Отсутствие неавторизованного доступа к информации
INTEGRITY **ЦЕЛОСТНОСТЬ**	Отсутствие некорректных изменений состояния системы
MAINTAINABILITY **ПОДДЕРЖИВАЕМОСТЬ**	Способность к восстановлению и модификации
ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ	Конфиденциальность + целостность + доступность
FAULT **НЕИСПРАВНОСТЬ**	Переход из состояния корректного выполнения функции в состояние некорректного выполнения функции
ERROR **ОШИБКА**	Следствие неисправности, часть состояния системы, при котором проявляется сбой
FAILURE **СБОЙ**	Ситуация, когда ошибка приводит к существенному изменению функционирования системы в состояние, не соответствующему спецификации

Немного теории: безотказность

Вероятность безотказной работы – это критерий, который достаточно полно характеризует надежность системы и может быть определен на стадии проектирования по вероятности безотказной работы составляющих ее элементов(компонентов).

Вероятность безотказной работы группы взаимосвязанных элементов – определяется как произведение вероятностей безотказной работы каждого элемента в этой группе.

$$P(t) = P_1(t) \cdot P_2(t) \cdot \ldots \cdot P_n(t) = \prod_{k=1}^{n} P_k(t)$$

Следствия:

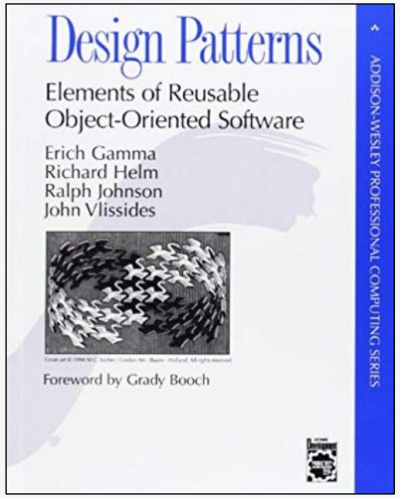
- Keep It Simple Stupid (K.I.S.S.)
- Бритва Оккама (“Не умножайте сущностей сверх необходимого”)

Обеспечение надежности на этапе проектирования

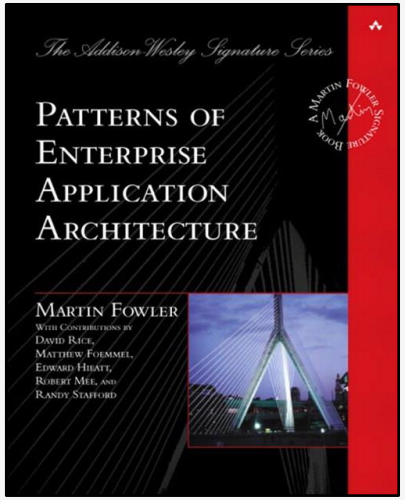
Общие принципы:

- Один элемент — одна функция, не создаем «всемогуторов»
- Максимально использовать простые структуры данных
- Необходимость учета взаимодействия элементов между собой в понятном и отлаживаемом формате
- Сложные алгоритмы использовать только если нет возможности использовать простые
- Не изобретаем велосипед, используем шаблоны проектирования и уже имеющиеся элементы
- Уменьшение функциональной избыточности (в системе нет элементов, которые не выполняют нужных функций)
- Изоляция компонентов системы, надежность и/или безопасность которых низка или не известна
- Структурная избыточность (когда применимо)

Книги:



GOF
«Design Patterns»

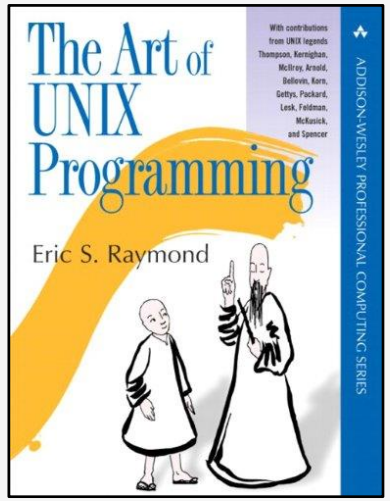


Martin Fowler
PoEAA

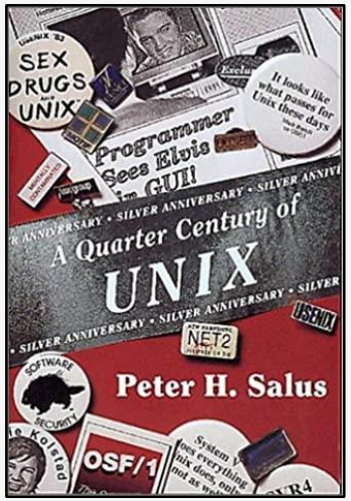
Практические примеры архитектурных решений

- OS UNIX (“UNIX way”) — pipes, утилиты, etc.
- Монолитные, модульные и микроядра
- chroot/jail, контейнеры, виртуальные машины
- Концепция Library OS
- Резервирование(кластеры), RAID

Книги:



Eric Raymond
«Art of UNIX programming»



Peter H. Salus
«Quarter Century of UNIX»

Надежность на этапе разработки

Fault prevention:

- Secure Development Lifecycle
  - Статический анализ кода
  - Модульное тестирование
  - Динамический анализ — как разрабатываемых, так и заимствованных компонентов
  - Автоматизация и инструменты везде, где возможно
- Сборочные кросс-платформенные системы
- Движение от простого к сложному, тестирование по мере реализации алгоритма

Fault tolerance:

- Обработка ошибок и исключений в языках программирования

Maintainability:

- Reproducible Builds

Надежность на этапе разработки: примеры

Системы сборки:

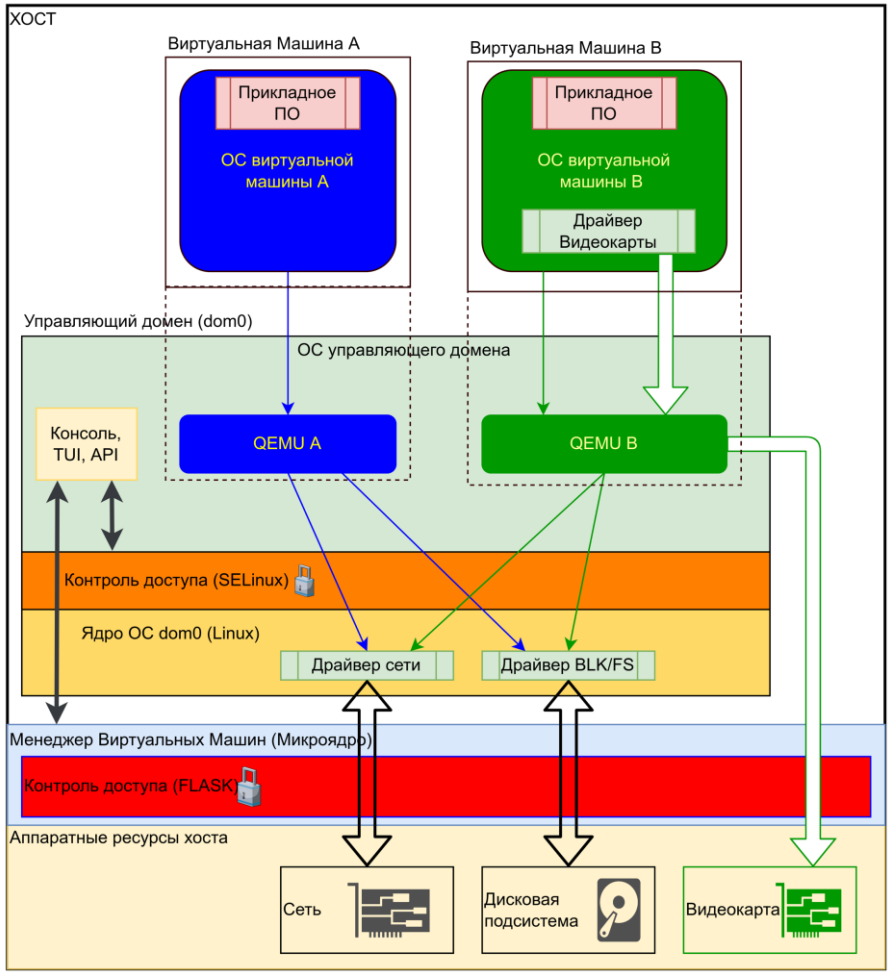
- OpenEmbedded: TanoWRT, OpenXT
- BuildRoot: «Синергия-Гипервизор»

Антипример:

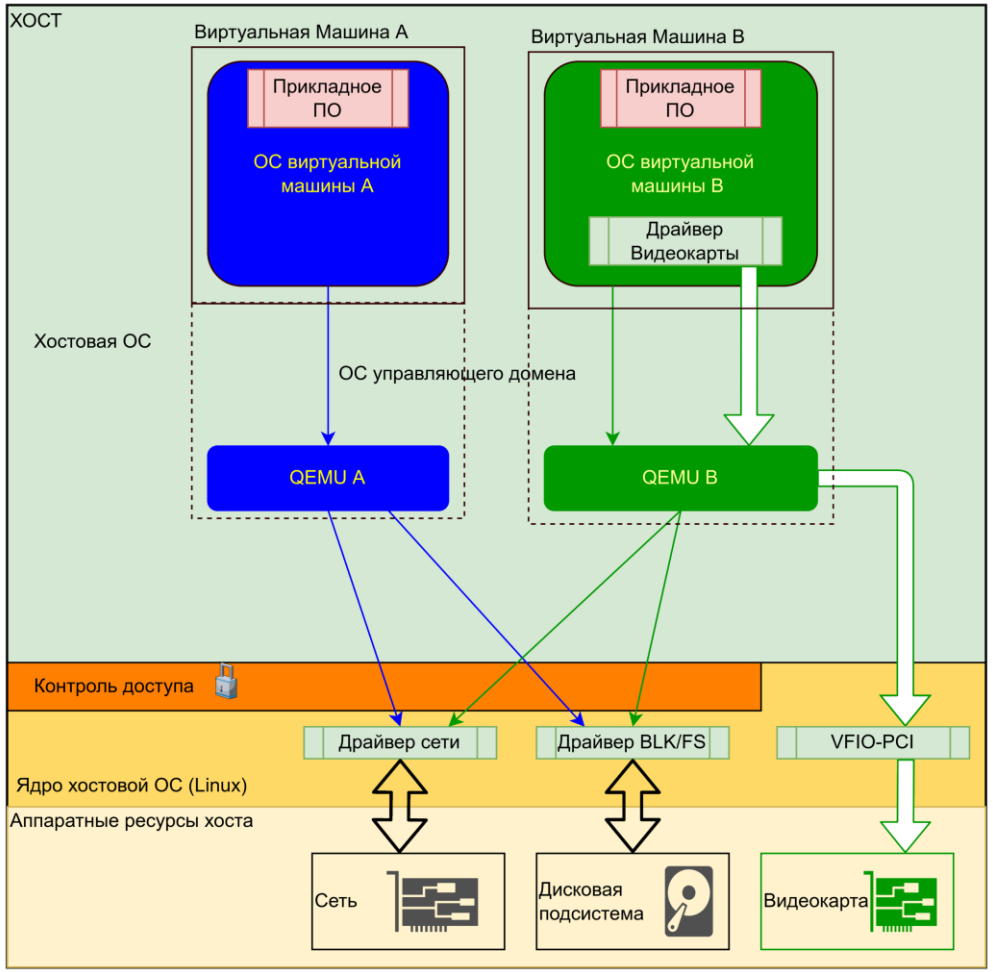
- OBS — Open Build System

Обеспечение надежности с использованием виртуализации

Варианты виртуализации: «виртуализация» на уровне ОС (контейнеры), гипервизоры 1 и 2 типа на примере XEN:



Гипервизор 1 типа



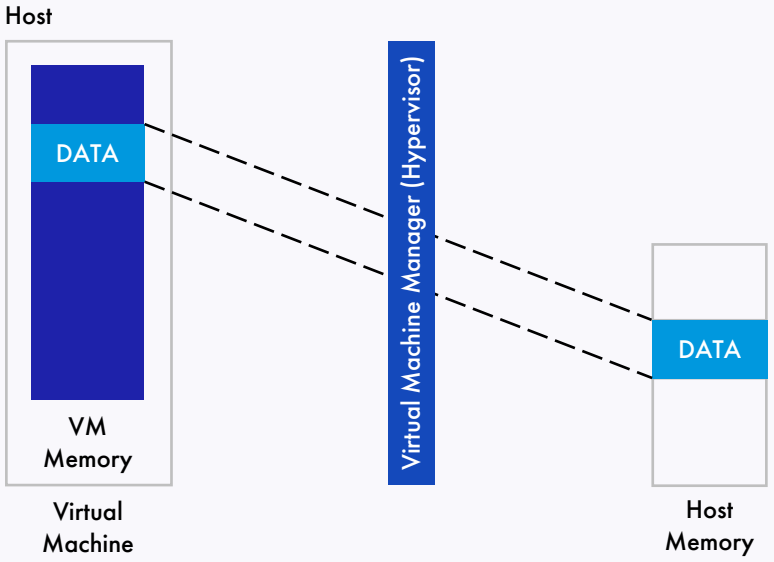
Гипервизор 2 типа

Контроль целостности

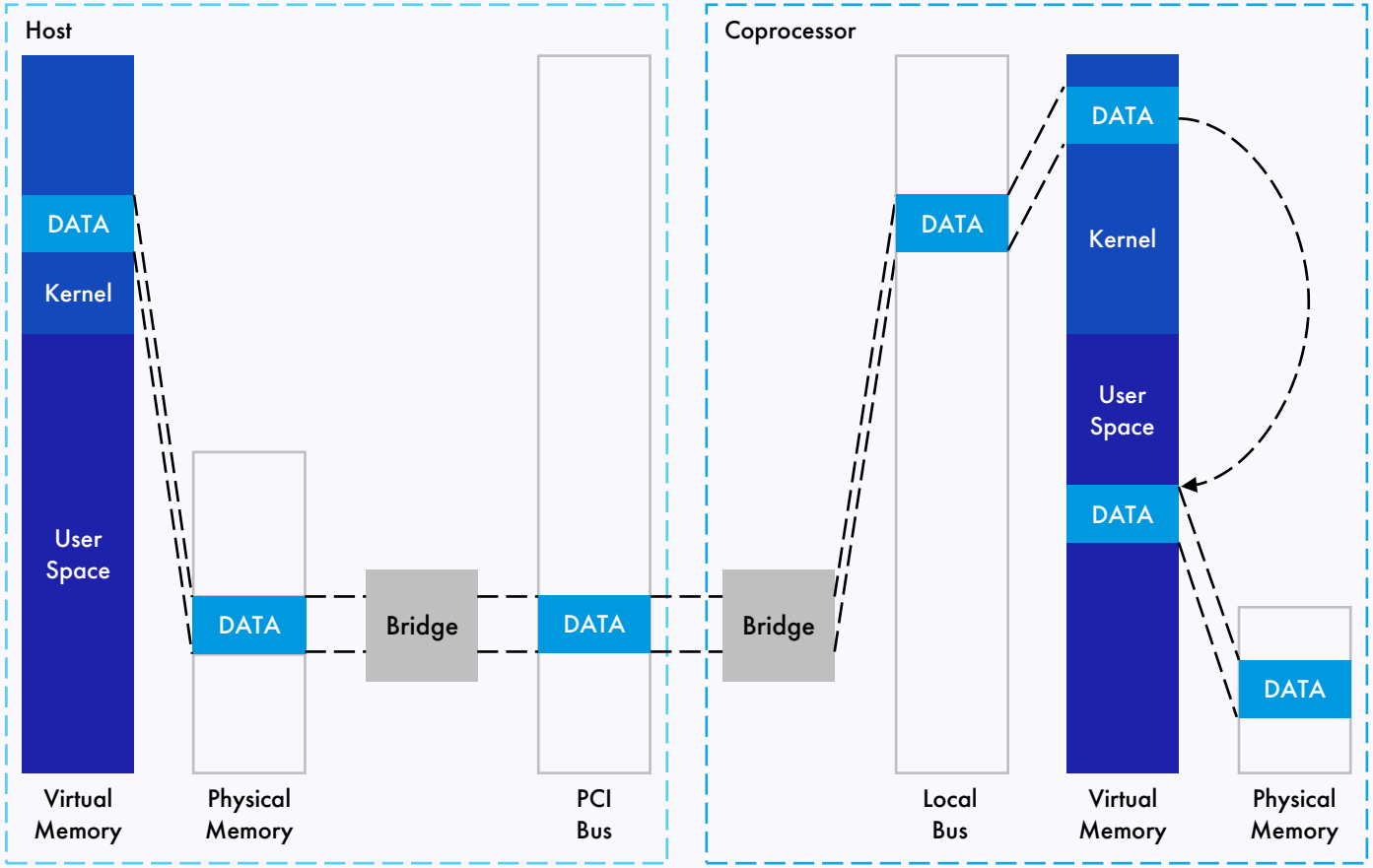
Примеры реализации:

- AIDE (userspace)
- auditd (условно, для maintainability/security)
- IMA/EVM (kernel)
- Мониторинг со стороны гипервизора (hardware — CPU)
- Мониторинг со стороны сопроцессора (external hardware)

Мониторинг со стороны гипервизора



Мониторинг со стороны сопроцессора*



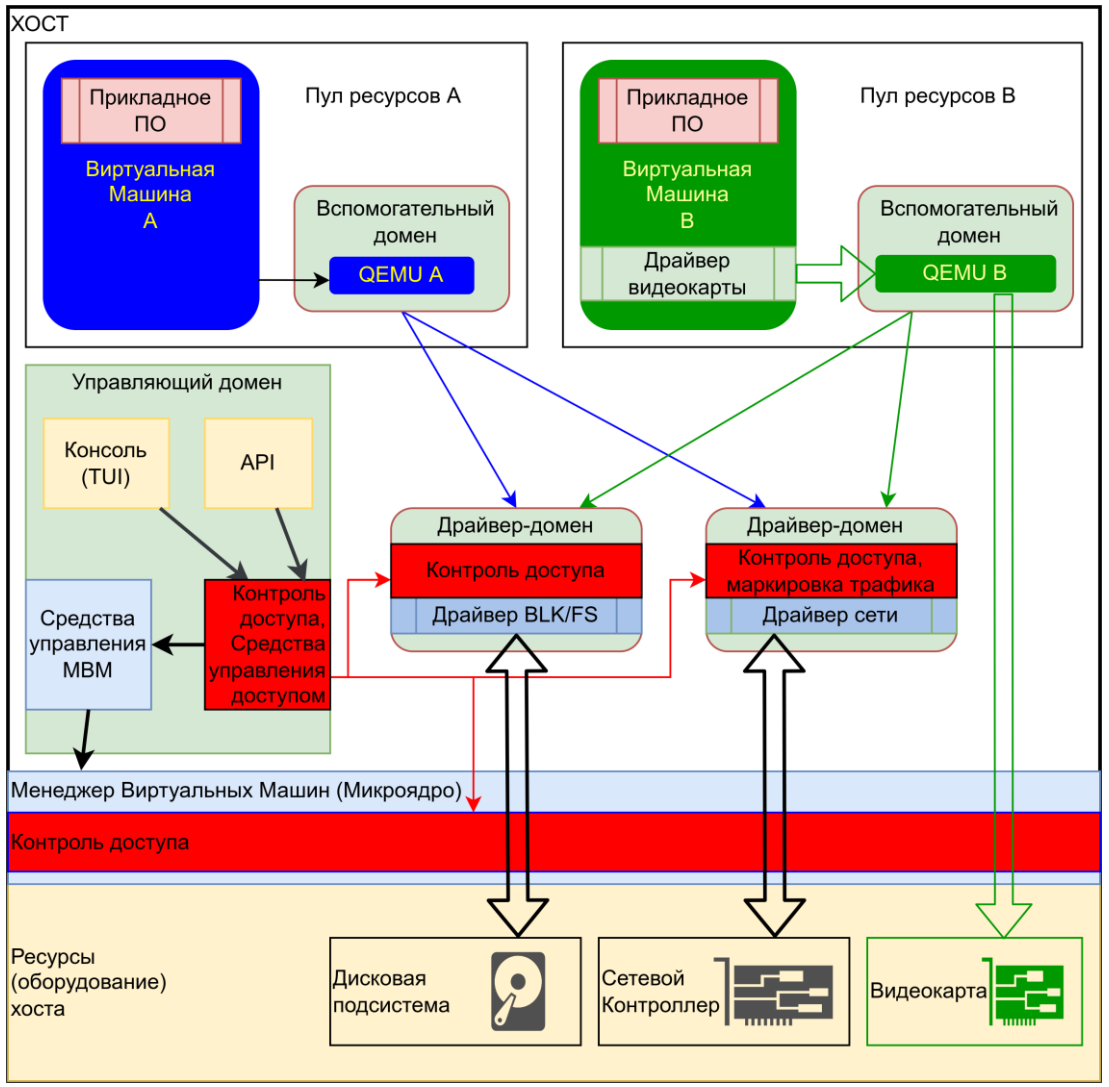
* **Источник:** Ph.D. dissertation of Nick Louis Petroni, Jr., «Property-based integrity monitoring of operating system kernels»

Изоляция и разделение ввода вывода

Один из способов увеличения надежности и восстанавливаемости системы —ослабление связей между объектами системы (или изоляция).

Примеры реализации:

- «Синергия-Гипервизор»
- Qubes OS (www.qubes-os.org)
- XOAR (www.cs.ubc.ca/~andy/papers/xoar-sosp-final.pdf)

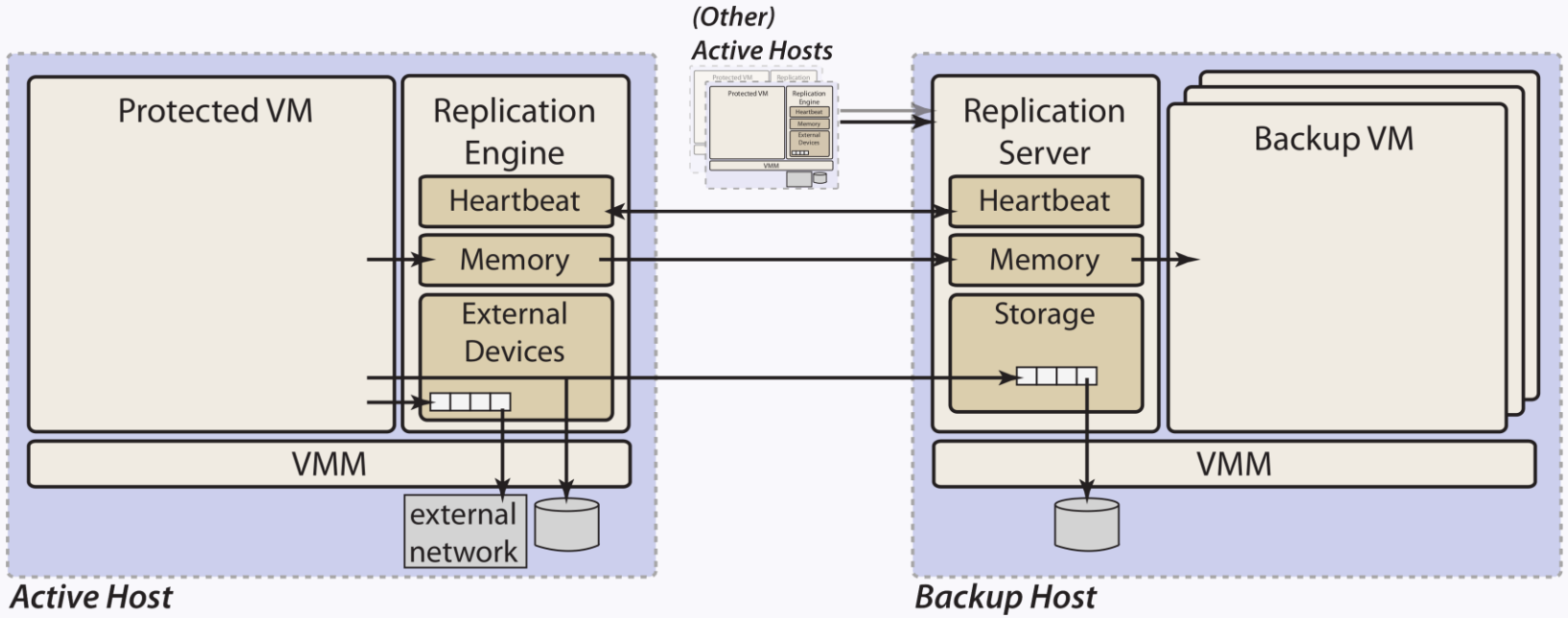


Архитектура гипервизора с выделенными доменами ввода-вывода

Виртуализация и FT/HA

Fault Tolerance — REMUS/COLO:

- Контроль доступа к памяти позволяет реплицировать изменения на резервный хост
- Репликация блочного устройства (диска) через DRBD (10Gbit direct Ethernet/40Gbit Infiniband)



High Availability:

- Общее дисковое устройство (iSCSI/FC/DRBD)
- Запуск ВМ «с нуля» на резервном хосте

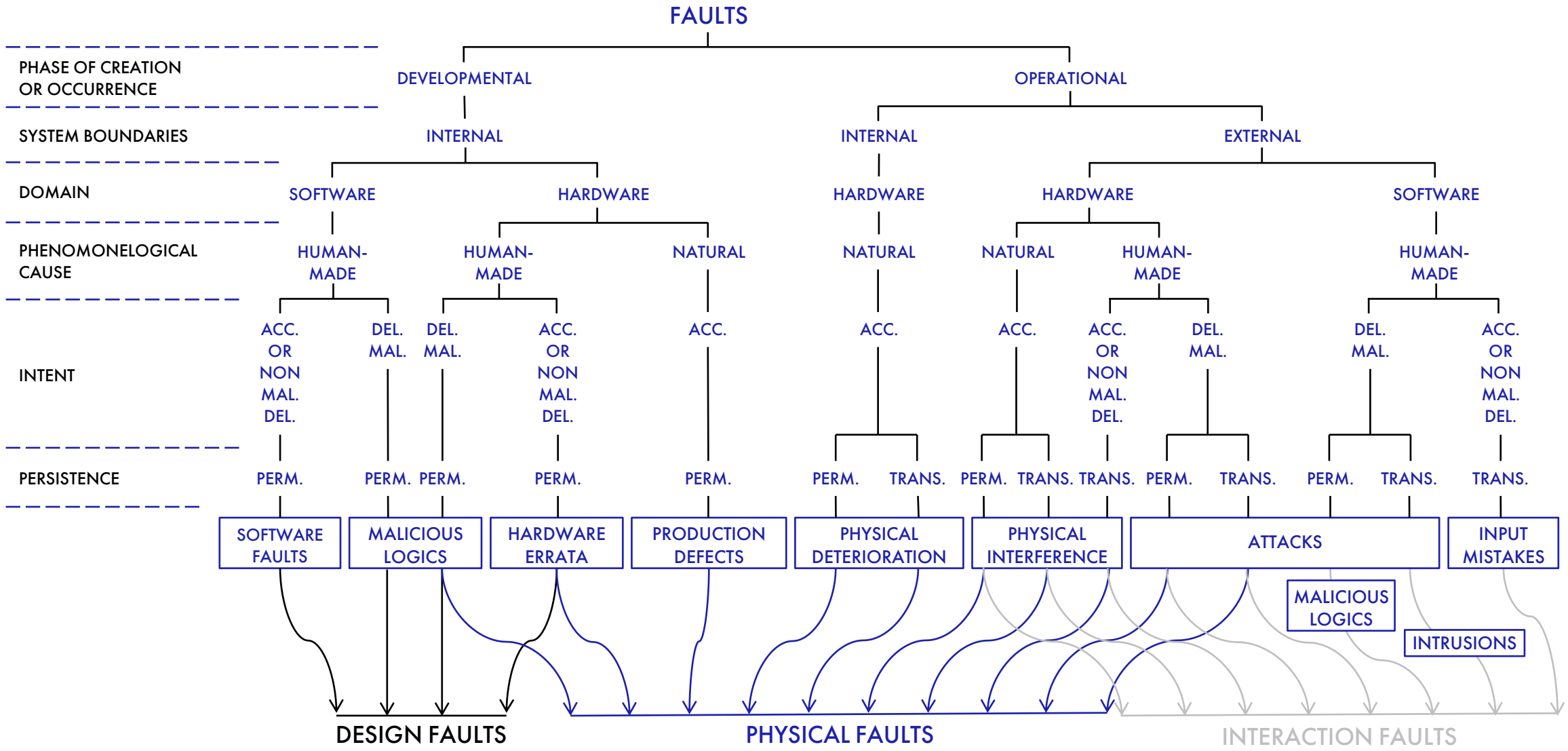
Заключение

- «Новое — хорошо забытое старое»
- Множество материалов по теории надежности, с очень полезными идеями
- Серия ГОСТов «Надежность в технике» частично применима и для ПО
- Полезные идеи в работах по безопасности и надежности с использованием виртуализации
- Огромное пространство для дальнейшего развития, особенно с учетом возможности реализации аппаратных механизмов в RISC-V

Литература и материалы

- «**Dependable Software, to appear in Dependable Systems: Software, Computing, Networks**»,
  Bertrand Meyer, eds. Jürg Kohlas, Bertrand Meyer, André Schiper, Lecture Notes in Computer Science, Springer-Verlag, 2006
- «**Fundamental Concepts of Dependability**», Algirdas Avizienis, Jean-claude Laprie, Brian Randell, September 2001
  (www.researchgate.net/publication/2408079_Fundamental_Concepts_of_Dependability)
- **«Predicting Dependability by Testing»**,
  Dick Hamlet, Portland State University, Department of Computer Science, Center for Software Quality Research, May 24, 1995
- **«Software Fault-Tolerance Techniques from a Real-Time Systems Point of View»**,
  Martin Hiller, Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Swedenc
- **«A method of vulnerability analysis based on deep learning for open source software»**, Yoshinobu Tamura and Shigeru Yamada,
  Tokyo City University, Tamazutsumi 1-28-1, Setagaya-ku, Tokyo 158-8557, Japan; Tottori Univerity, Minami 4-101, Koyama, Tottori-shi, 680-8552 Japan
- **«Breaking Up is Hard to Do: Security and Functionality in a Commodity Hypervisor»**, Patrick Colp, Mihir Nanavati, Jun Zhu , William Aiello, George Coker, Tim Deegan, Peter Loscocco and Andrew Warfield, Department of Computer Science, University of British Columbia, Citrix Systems R&D, NSA
- **«Property-based integrity monitoring of operating system kernels»**,
  Ph.D. dissertation of Nick Louis Petroni, Jr., 2008, Faculty of the Graduate School of the University of Maryland
- **«Remus: High Availability via Asynchronous Virtual Machine Replication»**, Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley,
  Norm Hutchinson, and Andrew Warfield, Department of Computer Science, University of British Columbia
- **«SecondSite: Disaster Tolerance as a Service»**,
  Shriram Rajagopalan, Brendan Cully, , Ryan O'Connor, Andrew Warfield, Department of Computer Science, University of British Columbia

Классификация ошибок и сбоев



yadro.com

123376, Москва г., Рочдельская ул., дом 15, строение 15

a.dubinin@yadro.com