TRANSFORMATION IN THE ENTERPRISE USING OPEN SOURCE SOFTWARE
WHY APPLICATION MODERNIZATION?
DIGITAL TRANSFORMATION DRIVES FLEXIBILITY AND OPENNESS

Most important technology area to the organization

- Create digital capability: 28%
- Modernize legacy systems: 23%
- Manage security, identity, and privacy: 18%
- Build the modern workplace: 15%
- Adopt cloud services: 10%

Vertically scalable, highly monolithical enterprise applications based on a highly integratted, proprietary HW/SW stack:

- Historically grown
- Usually critical business logic
- Written mostly in „old“ programming languages (e.g. COBOL, PL/1)
- Strong dependency on middleware stack (e.g. TPM)
- Lack of documentation
- Running not only (IBM) mainframes, but similar platforms from other vendors (e.g. BS/2000, Tuxedo, …)

**WHAT ARE „LEGACY SYSTEMS“? TRADITIONAL MAINFRAME-STYLE APPLICATIONS**

12.03.20
TRANSFORMATION USING OPEN SOURCE SOFTWARE | PROF. DR. PHILIPP BRUNE
WHAT IS „MODERN“?
TYPICAL OBJECTIVES

• Flexibility
  – Agile Development, DevOps

• Modern Application Architecture
  – Cloud-Ready/-Native (Microservices, Container, Serverless, ..)
  – Modern Languages and Frameworks
  – Open API

• Cost Efficiency
  – Flexible operating models (Cloud, On-Premise, Hybrid)
  – Open standards, Open Source Software (avoid vendor lock-in)
  – Risk reduction (e.g. regarding skill gap)
EXISTING APPROACHES:

- **Modernize existing applications on the mainframe platform** (e.g. wrapping COBOL transaction programs by web facades) (Sneed, 2001; Lee et al., 2001; Calladine, 2004; Ferguson and Stockton, 2005; Rodriguez et al., 2013; Mateos et al., 2017)

- **Extraction of business rules and logic** from existing COBOL code and subsequent re-implementation on other platforms (academics like this 😊) (Sneed, 1992; Huang et al., 1998; Sneed, 2001; Bodhuin et al., 2002; Lancia et al., 2007; Suganuma et al., 2008; Zhou et al., 2010; Mainetti et al., 2012; El Beggar et al., 2014)

- **Re-hosting** and subsequent modernization of COBOL applications on non-mainframe platforms, e.g. in the cloud (Talati and Lackie, 1999; Apte et al., 2017; http://www.lzlabs.com)
Examples of OSS technologies for modernizing applications:

- **Zowe** (https://www.zowe.org/)
  - RESTful API tools to access z/OS
  - Web GUI and CLI
  - Written in JavaScript / Node.js / Java
- **MS Visual Studio Code with IBM Z Open Editor plugin**
- **z/OS Container Extensions (z/CX)**
  - Run standard OCI (Docker) container images on z/OS
- **IBM Liberty Jakarta EE Application Server plus z/OS Connect including SwaggerUI integration**
- **Linux on Z Systems and LinuxONE**
  - IBM HyperProtect Cloud Services and Accelerator Program für Start-Ups...
AN OPEN SOURCE-BASED APPROACH...
QUICK WEB-BASED INTERACTIVE COBOL SERVICE (QWICS)

An Open Source Software framework for running transactional COBOL or C programs in the context of a Java / Jakarta EE app server (https://qwics.org)

- Use Jakarta EE Application Server as TPM replacement
- Avoid vendor lock-in with proprietary TPM emulation
- Modernize applications on different platforms (using Linux)
- Mix Java, JavaScript and COBOL
- Use only mature, enterprise-class OSS components:
  - GnuCOBOL compiler
  - OpenLiberty or JBoss WildFly
  - PostgreSQL
  - OpenJ9 JVM

ADVANTAGES

- Redesign your software architecture
  - Modernization, not Re-Hosting
  - Not a proprietary software emulation environment
- OSS as a HW abstraction layer
  - Runs on x86, IBM Z, POWER, etc.
  - Choose adequate platform for business case
- Avoid vendor lock-in

HOW DOES IT WORK?
RUN COBOL OR C CODE IN JAKARTA EE APP

Client — OpenLiberty or WildFly — EJB / WS — QWICS
COBOL / C Transaction Program — PostgreSQL

LEARN MORE
https://qwics.org

CONTACT
info@qwics.org
Twitter: @QWICS
LinkedIn:
https://de.linkedin.com/company/qwics
REFERENCES (1)


