Architectural Patterns/Styles

1. Please choose 2 projects

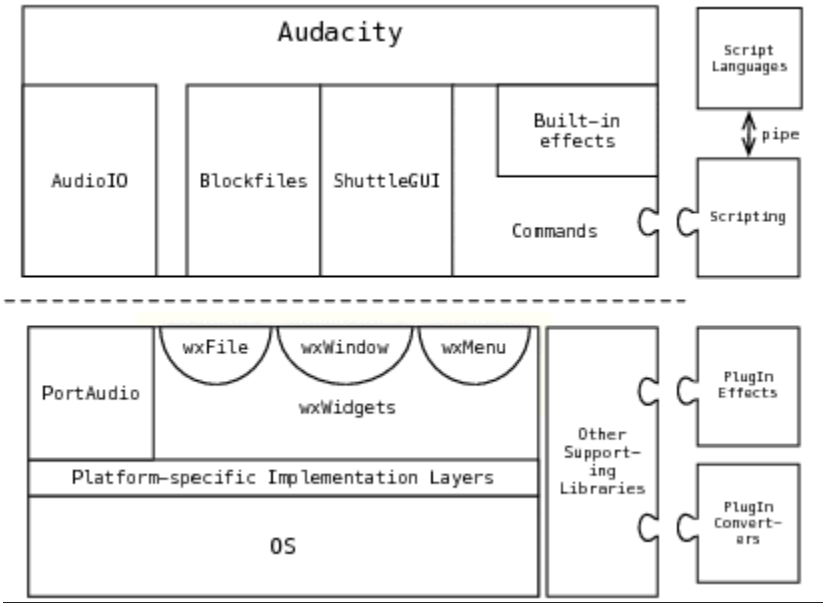
Audacity

purpose of the project

ต้องการสร้างแพลตฟอร์มเพื่อพัฒนาและแก้ปัญหาอัลกอริธึมการประมวลผลเสียง

architectural patterns

Audacity ใช้ architectural patterns แบบ Microkernel (Plug-in) มีส่วน core system และรองรับ plug-in ส่วน core system ของ Audacity ใช้รูปแบบ layer



quality attribute scenarios.

- Performance : Audacity มีการแบ่งไฟล์เสียงเป็น block file หลายๆไฟล์และมีการเชื่อมโยงกัน
  ต้องการแทรกบางอย่างลงไฟล์เสียงที่มีขนาดใหญ่
  ไม่ได้แบ่งเป็นไฟล์ record ที่มีขนาดใหญ่ : ต้องมีการคัดลอกและย้ายข้อมูลใน memory จำนวนมากทำให้ Audacity มีการทำงานที่ช้า
  แบ่งไฟล์ : จะช่วยให้ไม่ต้องมีการคัดลอกไฟล์จำนวนมากแต่เปลี่ยนการเชื่อมโยงแทนทำให้มีความเร็วที่มากขึ้น

- Portability : มีปลั๊กอินที่รองรับภาษาสคริปต์ได้หลายภาษา
  User → สั่งงานด้วยภาษาสคริปที่หลากหลาย → audacity สามารถทำงานได้
  User หลายคนที่ใช้ภาษาสคริปต์ต่างภาษา → drive audacity → สามารถทำงานได้

- Usability : Audacity มีการพัฒนา GUI ขึ้นเพื่อให้ง่ายต่อการใช้งาน
  User → ใช้งานผ่าน GUI → สามารถเข้าใจและใช้งานได้ง่ายขึ้น

GPSD

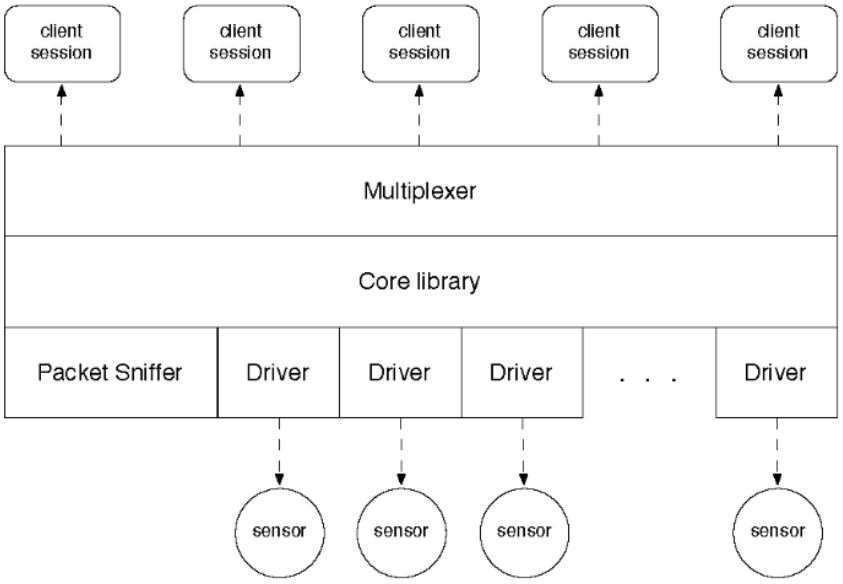
purpose of the project

เนื่องจากเซ็นเซอร์ที่เกี่ยวข้องกับการนำทางอื่นๆ ได้รับการออกแบบมาไม่ดีและมีความผันแปรสูงตามประเภทและรุ่นของเซ็นเซอร์

GPSD มีเพื่อซ่อนความอัปลักษณ์ที่ขึ้นกับอุปกรณ์ทั้งหมดไว้เบื้องหลังอินเทอร์เฟซไคลเอ็นต์

architectural patterns

แบบ layer



quality attribute scenarios.

Performance (resource utilization) : ตอนที่พัฒนา embedded system เน้นใช้ปริมาณหน่วยความจำและ CPU ต่ำ

เมื่อต้องการใช้ GPSD บนระบบขนาดเล็ก → สามารถใช้งานได้

Portability : สามารถติดตั้งบนระบบขนาดเล็กได้

Modifiability : เปลี่ยนจาก protocol รายงานเดิมไปใช้ JSON
เมื่อต้องการตรวจสอบ → ง่ายต่อการตรวจสอบ → ง่ายต่อการแก้ไข

Usability : GPSD พัฒนาโดยเน้นให้สามารถใช้งานได้เลยโดยไม่ต้องปรับค่า Application ที่ทราบตำแหน่ง
เมื่อนักพัฒนาต้องการใช้ GPSD → นำไปใช้ได้เลย → ไม่ต้องมีการปรับค่า Application

2. Please choose 1 project

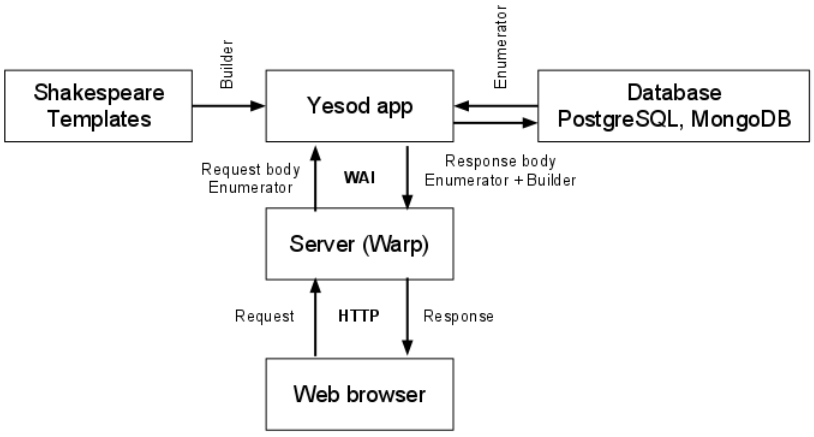
Yesod

purpose of the project

การขยายจุดแข็งของ Haskell ไปสู่การพัฒนาเว็บ Yesod เน้นให้โค้ดกระชับที่สุดตรวจสอบโค้ดทุกบรรทัด ตอนเวลาคอมไพล์ให้มากที่สุดเท่าที่เป็นไปได้ แทนที่จะต้องใช้ไลบรารีขนาดใหญ่

architectural patterns

client-server & N-Tier



quality attribute scenarios.

Performance : ลดการเรียกใช้ระบบและลดการสำนำบัฟเฟอร์ให้น้อยที่สุด

Buffer ไม่มาก → หน่วยความจำเหลือใช้ → application จะไม่ช้าลง

Security : การทดสอบ WAI

Req → WAI test → access

fake req → WAI test → not access

Reliability : มีการใช้กฎจำนวนมากในฐานข้อมูลเพื่อป้องกันไม่ให้ข้อมูลสูญหายและแจ้งข้อมูลความผิดพลาดกลับ

เกิดการผิดพลาด → กฎข้อกำหนดป้องกันข้อมูล → ข้อมูลจะไม่สูญหาย