Matplotlib

Matplotlib เป็นไลบรารี่สำหรับการพล็อตโดยมีภาษาไพธอนเป็นฐาน (Python-based plotting library) ด้วยกับการสนับสนุนอย่างเต็มที่สำหรับการพล็อตในรูปแบบสองมิติ (2D) อีกทั้งรองรับสำหรับการพล็อตกราฟฟิกสามมิติ (3D graphics) โดยใช้กันอย่างแพร่หลายในแวดวงของการคำนวณทางวิทยาศาสตร์ด้วยภาษาไพธอน เป้าหมายของไลบรารี่ Matplotlib มุ่งเป้าไปที่กรณีการใช้งานที่หลากหลาย มันสามารถที่จะฝั่งกราฟฟิกในชุดเครื่องมือการติดต่อกับผู้ใช้งาน (user interface toolkit) และตอนนี้รองรับกราฟฟิกที่สามารถตอบสนองได้บนระบบปฏิบัติการที่ใช้กันอย่างแพร่หลายด้วยชุดเครื่องมือ GTK+, Qt, Tk, FLTK, wxWidgets และ Cocoa มันสามารถที่จะเรียกใช้อย่างโต้ตอบผ่าน interactive Python shell เพื่อสร้างภาพกราฟฟิกด้วยคำสั่งขั้นตอนอย่างง่ายเช่น Mathematica, IDL และ MATLAB และมันสามารถที่จะฝั่งตัวใน headless webserver เพื่อจัดเตรียมฉบับพิมพ์ (hardcopy) ในรูปแบบ raster-based เช่น Portable Network Graphics (PNG) และรูปแบบเวกเตอร์เช่น PostScript และรูปแบบ Portable Document Format (PDF) และรูปแบบสุดท้าย Scalable Vector Graphics (SVG)

วัตถุประสงค์

Matplotlib เป็นไลบรารี่ที่ครอบครุมสำหรับการแสดงภาพ (visualization) ในรูปแบบคงที่ (static) เคลื่อนไหว (animated) โต้ตอบได้ (interactive) ด้วยภาษาไพธอน Matplotlib ช่วยให้ทำสิ่งเหล่านี้ให้ง่ายขึ้นและสะดวกยิ่งขึ้น

Architectural Patterns / Styles

รูปแบบสถาปัตยกรรมที่ Matplotlib ใช้เป็นรูปแบบ Layer architectural โดยมีทั้งหมดสามเลเยอร์

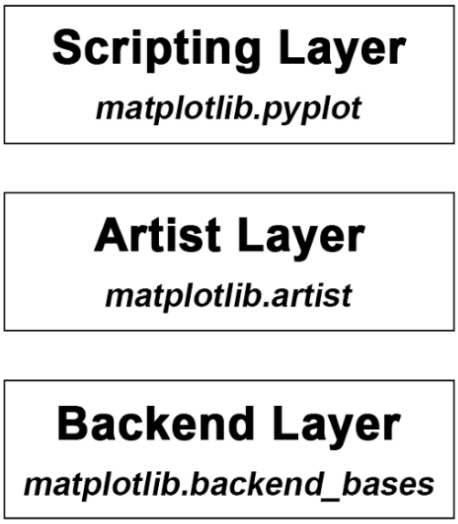
- Scripting Layer เป็นเลเยอร์ที่มีการเขียนสคริปต์ที่เบาที่สุด (Scripting interface) ในบรรดาเลเยอร์ทั้งสาม ถูกออกแบบมาเพื่อทำให้ไลบรารี่ Matplotlib ทำงานคล้ายคลึงกับ MATLAB script เป็นเลเยอร์ที่อยู่ชั้นบนสุด เป็นเลเยอร์ที่รวบรวมชุดคำสั่งและง่ายต่อการใช้งาน Artist Layer เป็นเลเยอร์ที่หนักในเชิงวากยสัมพันธ์ (syntactic) เนื่องจากมันถูกออกแบบว่าไว้สำหนับนักพัฒนาและไม่ได้ออกแบบมาไว้

สำหรับคนที่ต้องการศึกษาอย่างรวบรัดดังนั้น Scripting Layer เป็นเลเยอร์ที่ง่ายต่อการใช้งาน บางครั้งอาจจะเรียกว่า procedural plotting

- Artist Layer เป็นเลเยอร์ที่ช่วยให้ควบคุมและปรับจูนของ figure ให้ได้มากที่สุดเท่า เปรียบดั่งนักจิตรกรที่กำลังวาดบนผ้าใบ เลเยอร์นี้ประกอบไปด้วยหนึ่งออปเจ็คหลักคือ Artist ที่ใช้ Renderer เพื่อวาดภาพบนแคนวาส มันสามารถที่จะปรับแต่งได้มากว่าเมื่อเปรียบเทียบกับ Scripting Layer และมันสะดวกกว่าสำหรับการพล็อตขั้นสูง โดยเฉพาะอย่างยิ่งเมื่อจัดการเกี่ยวกับหลายรูปภาพหรือหลายแกนมันจะทำให้เราไม่สับสนว่าเรากำลังทำงานอยู่กับภาพหรือแกนไหนเนื่องจากทุก ๆ subplot ทั้งหมดถูกกำหนดค่าไปยังออปเจ็คของ Artist ทำให้บางครั้งสามารถที่เรียก Artist Layer ว่า object-based plotting ทุก ๆ สิ่งที่เห็นบน Matplotlib figure เป็นอินสแตนท์ของ artist เช่น Title, lines, tick labels, images เป็นต้น ออปเจ็คของ Artist มีอยู่ 2 ประเภท ได้แก่ primitive type เช่น line2d, rectangle, circle และ text รูปแบบที่สองคือ composite type เช่น Axis, Tick, Axes และ Figure

- Backend Layer เป็นเลเยอร์ที่จัดการงานที่หนักผ่านการสื่อสารไปยังชุดเครื่องมือวาดภาพในเครื่องเช่น wxPython หรือภาษาวาดภาพ (drawing language) อย่าง PostScript เป็นเลเยอร์ที่ซับซ้อนที่สุด ประกอบไปด้วยสามบิลท์อินคลาสหลัก ๆ (built-in abstract interface classes)
    1. FigureCanvas เป็นแคสวาสที่จะแสดงรูปภาพ
    2. Renderer เป็น abstract class ที่จัดการในเรื่องการวาดและการแสดงผล มีหน้าที่ในการวาด ใน FigureCanvas
    3. Event จัดการเกี่ยวการป้อนของผู้ใช้งานเช่น การกดคีย์บอร์ดและการกดเมาส์เป็นต้น

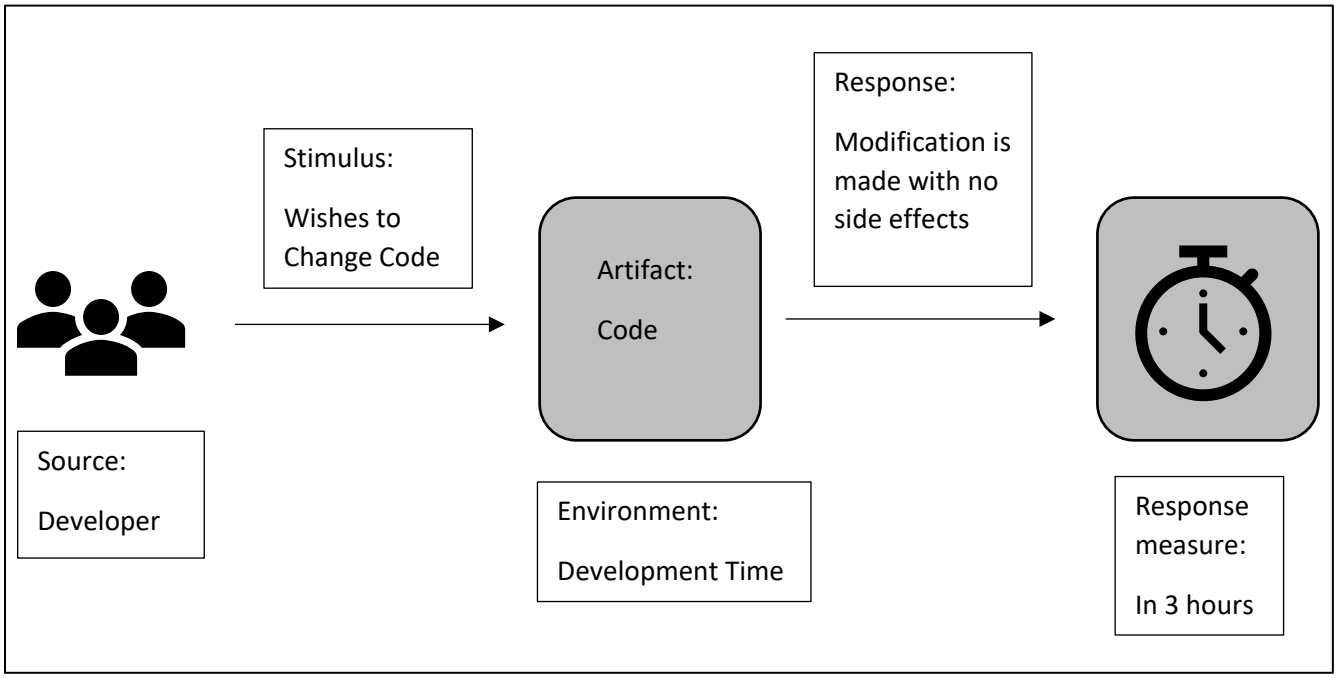
รูปภาพประกอบเพื่อให้เห็นภาพ



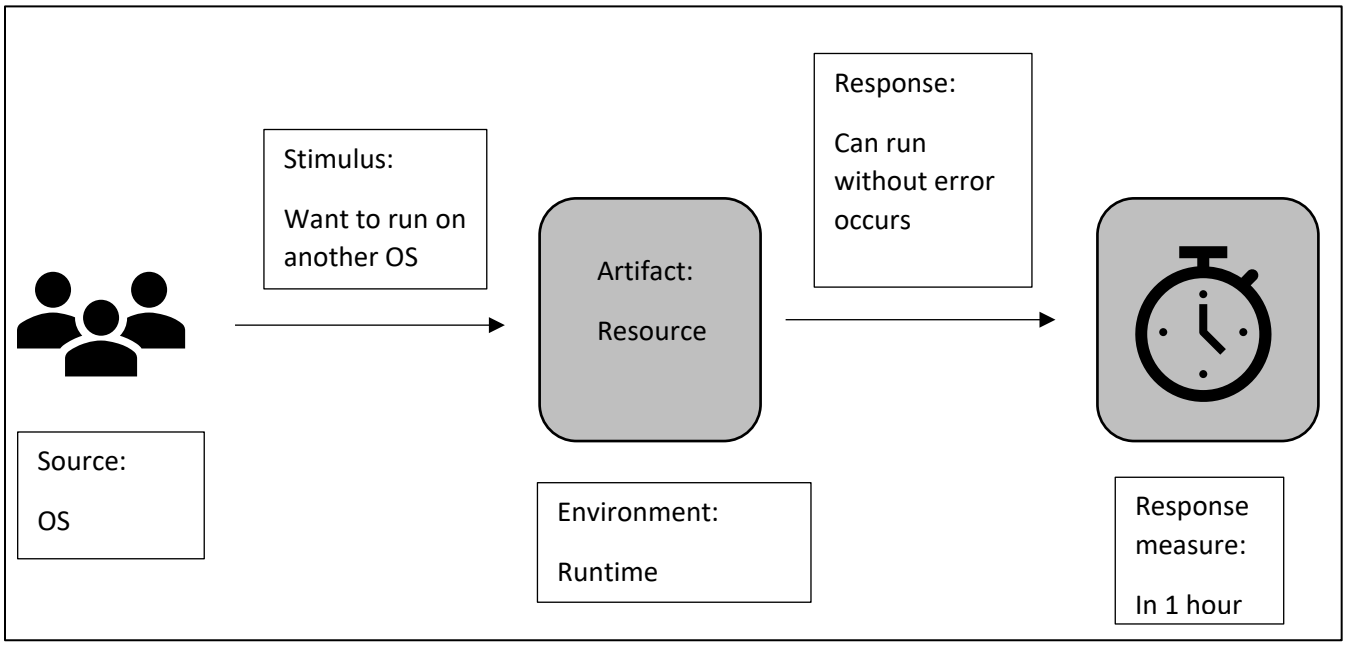
รูป 1 https://medium.datadriveninvestor.com/data-visualization-with-python-matplotlib-architecture-6b05af533569

Quality Attribute Scenarios

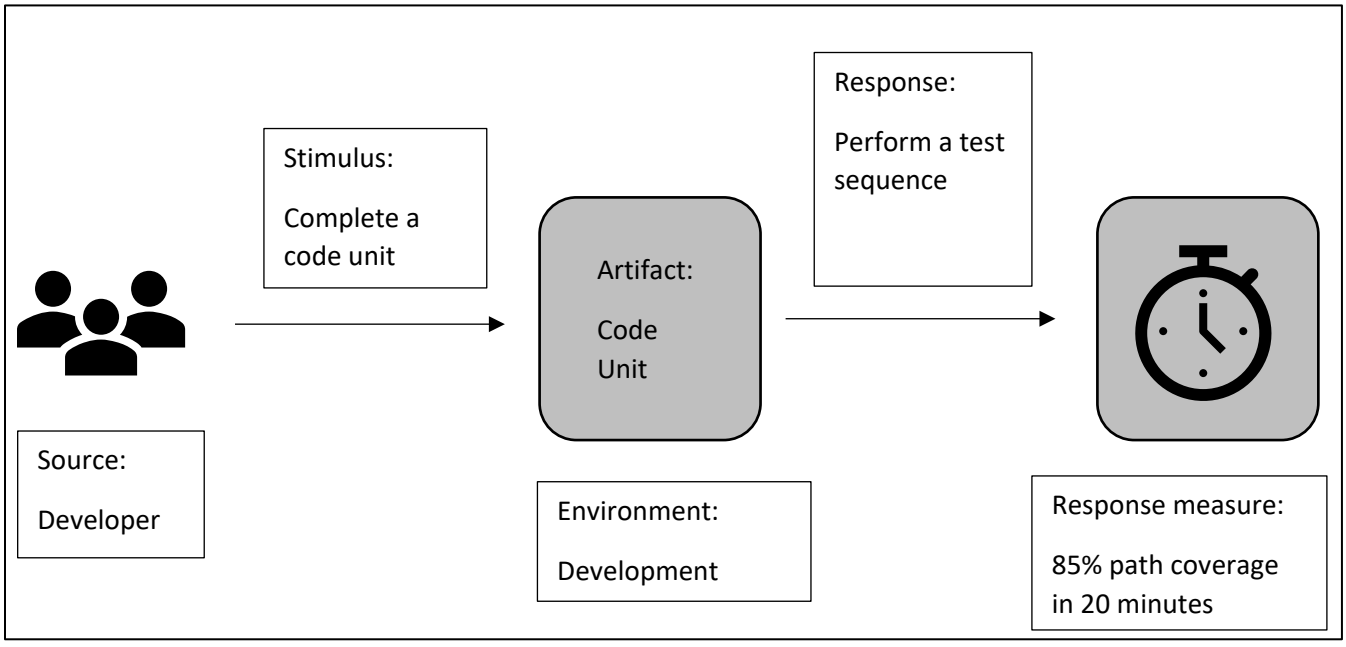
Modifiability



Portability



Testability



แหล่งอ้างอิง

- https://medium.datadriveninvestor.com/data-visualization-with-python-matplotlib-architecture-6b05af533569
- https://www.aosabook.org/en/matplotlib.html

GPSD

GPSD เป็นชุดเครื่องมือสำหรับจัดการคอลเลกชั่นของอุปกรณ์จีพีเอสและเซนเซอร์อื่น ๆ ที่เกี่ยวกับกับนำทางและการจับเวลาที่แม่นยำรวมไปถึงวิทยุทางทะเล AIS (Automatic Identification System) และเข็มทิศดิจิตอล โปรแกรมหลักคือ gpsd เป็นโปรแกรมที่ทำงานอยู่เบื้องหลัง (daemon) จัดการเกี่ยวกับชุดของเซนเซอร์และทำรายงานจากชุดเซนเซอร์ด้วยเจซัน (JSON) ออปเจ็คฉ่ายบน TCP/IP พอร์ต โปรแกรมอื่น ๆ ในชุด gpsd ประกอบด้วยการสาธิตไคลเอ็นต์ (demonstration client) ที่สามารถใช้เป็นโมเดลโค้ดและเครื่องมือวินิจฉัย

GPSD ถูกนำไปใช้อย่างกว้างขวางบนคอมพิวเตอร์แบบพกพก (Laptop) สมาร์ทโฟน ยานพาหนะอัตโนมัติรวมไปถึงรถยนต์ไร้คนขับและเรือดำน้ำ gpsd มีฟีเจอร์ที่เป็นระบบฝังตัวใช้สำหรับการนำทาง เกษตรแม่นยำ มาตรวิทยาที่ไวต่อตำแหน่ง และบริกาเวลาเครือข่ายอีกทั้งยังใช้ในระบบ Identification-Friend-or-Foe ของยานเกราะต่อสู้รวมถึง M1 “Abrams” รถถังต่อสู้หลัก

GPSD เป็นโปรเจ็คขนาดกลางประมาณ 43 KLOC (thousands of lines of code) โดยเขียนด้วยภาษาซีและไพธอนเป็นหลัก gpsd มีอัตราข้อบกพร่องที่ต่ำมากในอดีตโดยวัดทั้งเครื่องการตรวจสอบเช่น splint valgrind และ Coverity และอุบัติเหตุของรายงานข้อผิดพลาดของตัวติดตามและที่อื่น ๆ สิ่งนี้ไม่ได้บังเกิดขึ้นโดยบังเอิญ โปรเจ็คใช้ความพยายามอย่างมากในการผสมผสานเทคโนโลยีสำหรับการทดสอบโดยอัตโนมัติ และความพยายามนั้นก็ได้ผลดี

โปรแกรมหลักในชุดของ GPSD คือ gpsd service daemon มันสามารถรวบรวมจากชุดอุปกรณ์เซนเซอร์ผ่าน RS232 USB Bluetooth TCP/IP และ UPD links การรายงานโดยทั่วไปจะส่งไปยัง TCP/IP พอร์ต 2947 แต่สามารถที่จะออกผ่านหน่วยความจำแบ่งปัน (shared-memory) หรือ D-BUS interface gpds ส่งด้วยไคลเอนต์ไลบรารี่ด้วยภาษาซี ซีพลัสพลัส และ ไพธอน มันประกอบด้วยไคลเอนต์ในภาษาซี ซีพลัสพลัส ไพธอน และ พีเอชพี อย่างง่าย ไคลเอนต์ภาษาเพิร์ลผูกติดพร้อมใช้งานผ่าน CPAN ไคลเอนต์นี้ไม่ได้เป็นเพียงแค่ความสะดวกสำหรับนักพัฒนาแอปพลิเคชันเท่านั้น มันป้องกันคนที่พัฒนา GPSD จากการปวดหัวด้วย โดยแยกแอปพลิเคชันออกจากรายละเอียดของโปรโตคอลการรายงาน JSON ของ GPSD ดังนั้น API ที่เปิดเผยต่อไคลเอ็นต์จะยังคงเหมือนเดิมแม้ว่าโปรโตคอลจะขยายคุณลักษณะใหม่สำหรับเซ็นเซอร์ประเภทใหม่

โปรแกรมอื่นที่มาพร้อมชุม GPSD คือ gpsmon เป็นโปรแกรมสำหรับมอนิเตอร์อุปกรณ์ gpsprof เป็นโปรแกรมสำหรับสร้างโปรไฟล์ที่สร้างรายงานเกี่ยวกับสถิติข้อผิดพลาดและระยะเวลาของอุปกรณ์ gpsctl เป็นโปรแกรม

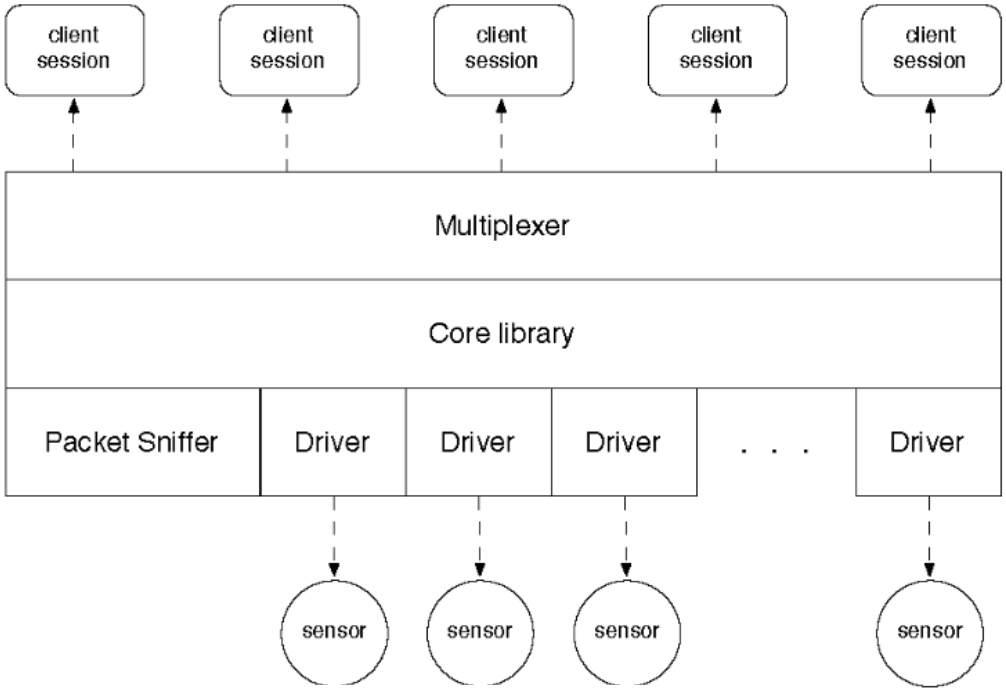
สำหรับปรับแต่งการตั้งค่าอุปกรณ์ gpsdecode เป็นโปรแกรมสำหรับเซ็นเซอร์การแปลงแบบแบตช์ (batch) จะบันทึกลงใน JSON ที่อ่านได้

วัตถุประสงค์

Gpsd เป็น service daemon ที่คอยมอนิเตอร์ GPS หรือ AIS receiver (Automatic Identification System) ที่ต่อกับโฮสต์คอมพิวเตอร์ผ่านพอร์ตอนุกรมหรือพอร์ต USB บน TCP 2947 ของโฮสต์คอมพิวเตอร์ GPSD ถูกนำไปใช้อย่างกว้างขวางบนคอมพิวเตอร์แบบพกพก (Laptop) สมาร์ทโฟน ยานพาหนะอัตโนมัติรวมไปถึงรถยนต์ไร้คนขับและเรือดำน้ำ gpsd มีฟีเจอร์ที่เป็นระบบฝังตัวใช้สำหรับการนำทาง เกษตรแม่นยำ มาตรวิทยาที่ไวต่อตำแหน่ง และบริกาเวลาเครือข่าย

Architectural Patterns / Styles

โดยสถาปัตยกรรมซอฟต์แวร์ gpsd ใช้คือ Microkernel (plug-in) โดยตรง core system สามารถที่แบ่งเป็น 4 องค์ประกอบได้แก่ Driver, Packet Sniffer, Core library, Multiplexer โดยองค์ประกอบจะเรียงกันเป็น Layer



รูป 2 https://aosabook.org/en/gpsd.html

Driver เป็น ไดรเวอร์อุปกรณ์ของผู้ใช้ (user-space device driver) สำหรับชิปเซ็ตเซ็นเซอร์ที่รองรับ

Packet Sniffer รับผิดชอบการขุดแพ็คเก็ตข้อมูลออกจากสตรีมอินพุตแบบอนุกรม โดยพื้นฐานแล้วมันเป็น state machine ที่คอยเฝ้าดูทุกสิ่งที่ดูเหมือนหนึ่งใน 20 ประเภทแพ็คเก็ตที่รู้จักโดยใช้รูปแบบ checksum ดังนั้นการระบุที่แม่นยำเป็นเรื่องที่ไม่ยาก เนื่องจากอุปกรณ์สามารถเสียบปลั๊กหรือเปลี่ยนโหมดได้ ประเภทของแพ็กเก็ตที่จะดึงมาจากพอร์ตอนุกรมหรือพอร์ต USB ไม่จำเป็นต้องได้รับการแก้ไขอย่างถาวรโดยตัวแรกที่รู้จัก

Core Library จัดการเซสชั่นด้วยอุปกรณ์เซ็นเซอร์ โดยมีลำดับการทำงานดังนี้

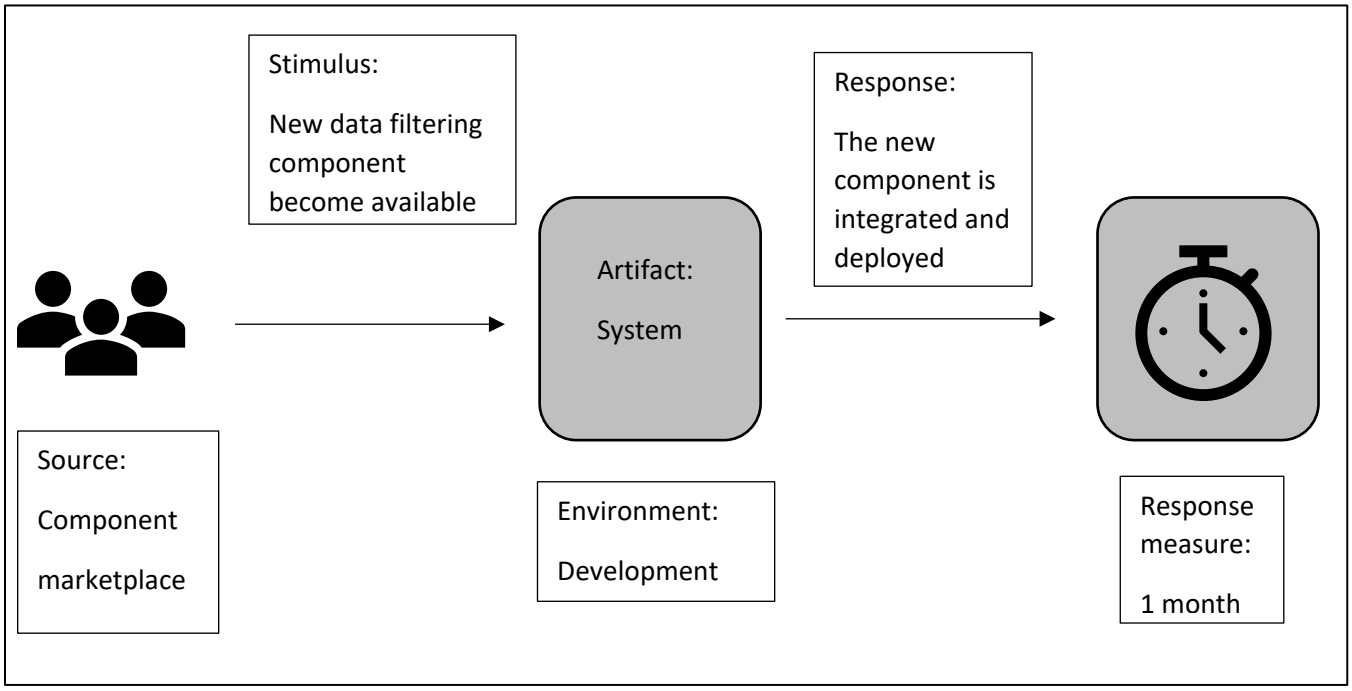
- เริ่มต้นเซสชั่นโดยเปิดอุปกรณ์และอ่านข้อมูลจากอุปกรณ์ ไล่ตามอัตราบอดและพาริตี/สต็อปบิตผสมกันจนกว่าแพริตีสนิฟเฟอร์จะล็อกการซิงโครไนซ์กับประเภทแพ็กเก็ตที่รู้จัก
- การสำรวจอุปกรณ์สำหรับแพ็กเก็ต
- ปิดอุปกรณ์และปิดเซสชั่น

ฟีเจอร์หลักของ core library คือมันทำหน้าที่สับเปลี่ยนการเชื่อมต่อ GPS ในการเชื่อมต่ออุปกรณ์ที่ถูกต้องขึ้นกับประเภทของแพ็กเก็จที่ sniffer ส่งค่ากลับ ไม่ได้กำหนดค่าไว้ล่วงหน้าและอาจเปลี่ยนแปลงได้เมื่อเวลาผ่านไป โดยเฉพาะอย่างยิ่งหากอุปกรณ์สลับไปมาระหว่างโปรโตคอลการรายงานต่าง ๆ

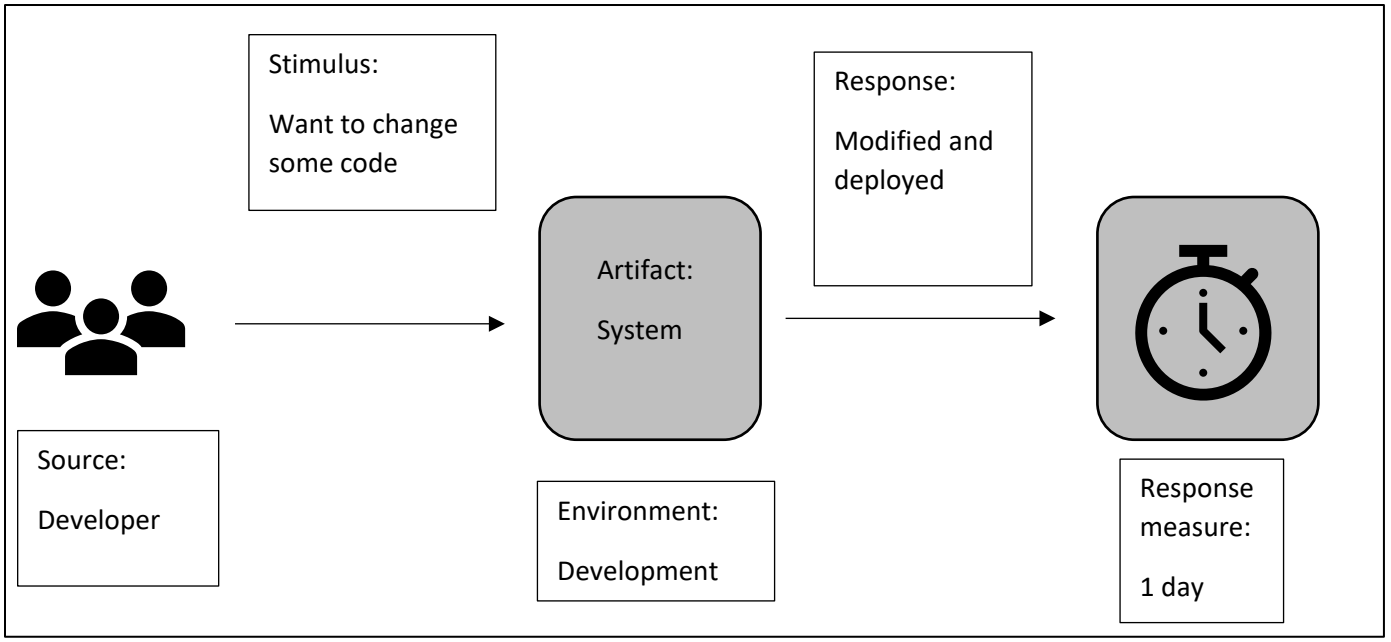
Multiplexer เป็นส่วนหนึ่งจาก daemon ที่ทำหน้าที่ client session และการกำหนดค่าให้กับอุปกรณ์ มันทำหน้าที่ส่งผ่านรายงานาให้กับ client และตอบรับคำสั่งจาก client และตอบสนองต่อ hotplug notification โดยพื้นฐานแล้วจะมีอยู่ในไฟล์ต้นฉบับหนึ่งไฟล์ gpsd.c และไม่เคยติดต่อกับไดรเวอร์อุปกรณ์โดยตรง องค์ประกอบอื่นที่นอกจาก Multiplexer ถูกเชื่องด้วยกันในไลบรารี่ libgpsd และสามารถใช้แยกจาก multiplexer ได้ เครื่องมืออื่น ๆ ที่ติดต่อกับเซ็นเซอร์โดยตรงเช่น gpsmon และ gpsctl โดยเรียกใช้ใน Core library และ Driver layer โดยตรง

Quality Attribute Scenarios

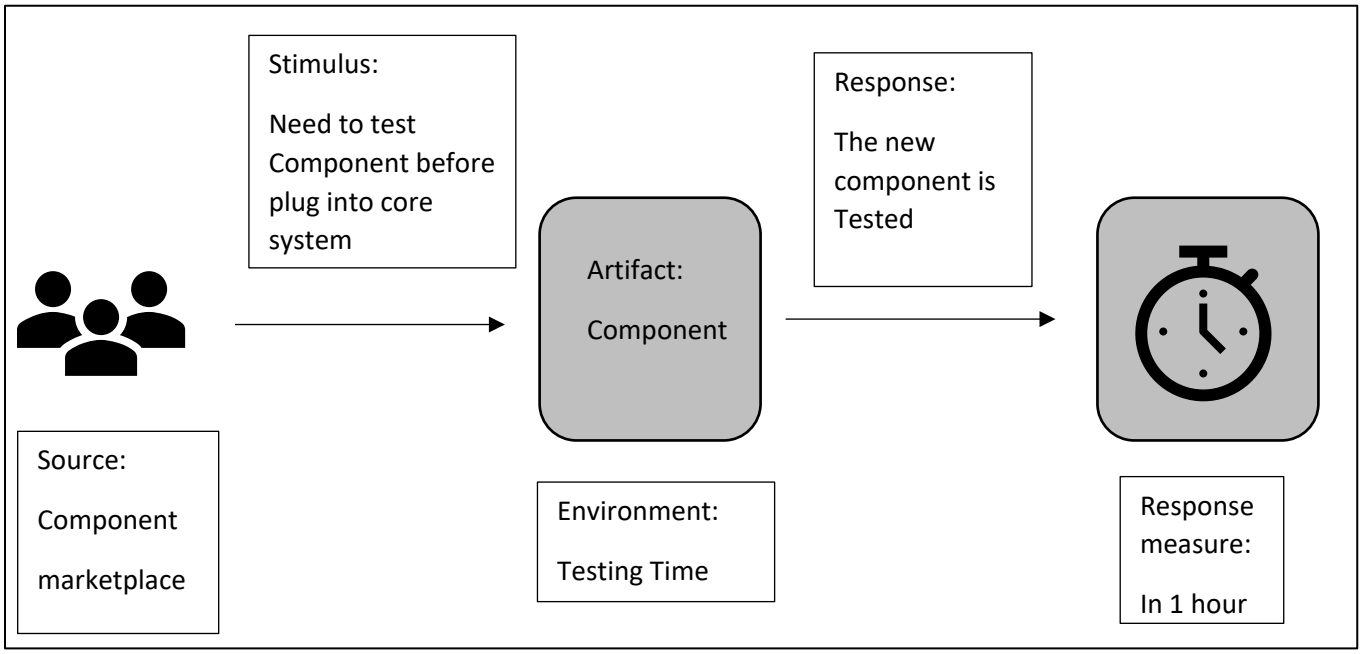
Integrability



Modifiability



Testability



แหล่งอ้างอิง

- https://aosabook.org/en/gpsd.html
- https://gpsd.io/

YESOD

Yesod เป็นเว็บเฟรมเวิร์คถูกเขียนด้วยภาษาฮาสเกล (Haskell programming language) ในขณะที่เว็บเฟรมเวิร์คที่โด่งดังใช้ประโยชน์จากลักษณะแบบไดนามิกของภาษาโฮสต์ yesod ใช้ประโยน์จากลักษณะที่เป็นรูปแบบคงที่ (static) ของภาษาฮาสเกลเพื่อให้การเขียนเป็นรูปแบบที่ปลอดภัยและเขียนโค้ดได้รวดเร็ว

การพัฒนาได้เริ่มต้นขึ้นมาตั้งแต่ปี ค.ศ. 2010 และก็ได้พัฒนามาเรื่อย ๆ yesod ได้ฟันฝ่าในโครงการชีวิตจริงด้วยกับฟีเจอร์ที่เริ่มต้นทั้งหมดที่เกิดจากของจริงความต้องการในชีวิตจริง แรกเริ่มการพัฒนา yesod เป็นในรูปแบบทำคนเดียวทั้งหมดหลังจากนั้นประหนึ่งปีให้หลังก็ได้มีกลุ่มคนคนได้เข้ามามีส่วนร่วมทำให้ yesod ได้เบ่งบานในโครงการ thriving open source project

ทำไมต้องใช้ภาษาฮาสเกล? ดูเหมือนว่าในโลกนี้จะมีรูปแบบสองแบบด้วยกัน

- Statically typed language เช่นภาษาจาวา ภาษาซีชาร์ป ภาษาซีพลัส ๆ ภาษาเหล่านี้ให้ความเร็วและความปลอยภัยของ type แต่ยากต่อการเขียนโค้ด
- Dynamically typed language เช่นภาษารูบี้ ภาษาไพธอน ภาษาเหล่านี้เพิ่มผลผลิตอย่างมากอย่างแต่รันได้ช้าและคอมไพเลอร์ไม่ค่อยสนับสนุนในการเช็คความถูกต้อง

คำตอบคือนี้เป็นแบ่งขั้วที่ผิด ไม่มีเหตุผลใดที่ว่าทำไมภาษาที่เป็น static typed ต้องเป็นภาษาที่งุ่มง่ามเชื่องช้า ฮาสเกลสามารถที่จะตรวจจับการแสดงออกจำนวนมากของ Ruby และ Python ในขณะที่ยังคงที่เป็น strongly typed language ในความเป็นจริงระบบของ Haskell type ตรวจจับบั๊คมากกว่าภาษาจาวาเช่น Null pointer exception ถูกกำจัดไว้เรียบร้อยแล้ว โครงสร้างข้อมูลที่ไม่เปลี่ยนรูปทำให้การให้เหตุผลเกี่ยวกับโค้ดของคุณง่ายขึ้น ลดความซับซ้อนของการเขียนโปรแกรมแบบขนานและพร้อมกัน

ทำไมต้องใช้ภาษาฮาสเกล? ฮาสเกลที่ภาษาที่มีประสิทธิภาพและภาษาที่เป็นมิตรกับนักพัฒนาซึ่งให้การตรวจสอบความถูกต้องของโปรแกรมคอมไพล์เวลารวบรวมมากมาย

วัตถุประสงค์

เป้าหมายของ Yesod คือการขยายจุดแข็งของ Haskell ไปสู่การพัฒนาเว็บ Yesod พยายามทำให้โค้ดของคุณกระชับที่สุด ทุกบรรทัดของโค้ดของคุณจะถูกตรวจสอบความถูกต้อง ณ เวลาคอมไพล์ให้มากที่สุดเท่าที่เป็นไปได้

แทนที่จะต้องใช้ไลบรารีขนาดใหญ่ของการทดสอบหน่วย (unit testing) เพื่อทดสอบคุณสมบัติพื้นฐาน คอมไพเลอร์ทำทุกอย่างเพื่อคุณ โดยเบื่องหลัง Yesod ใช้เทคนิคประสิทธิภาพขั้นสูงมากเท่าที่เราสามารถรวบรวม เพื่อให้โค้ดระดับสูงของคุณทำงาน

Architectural Patterns / Styles

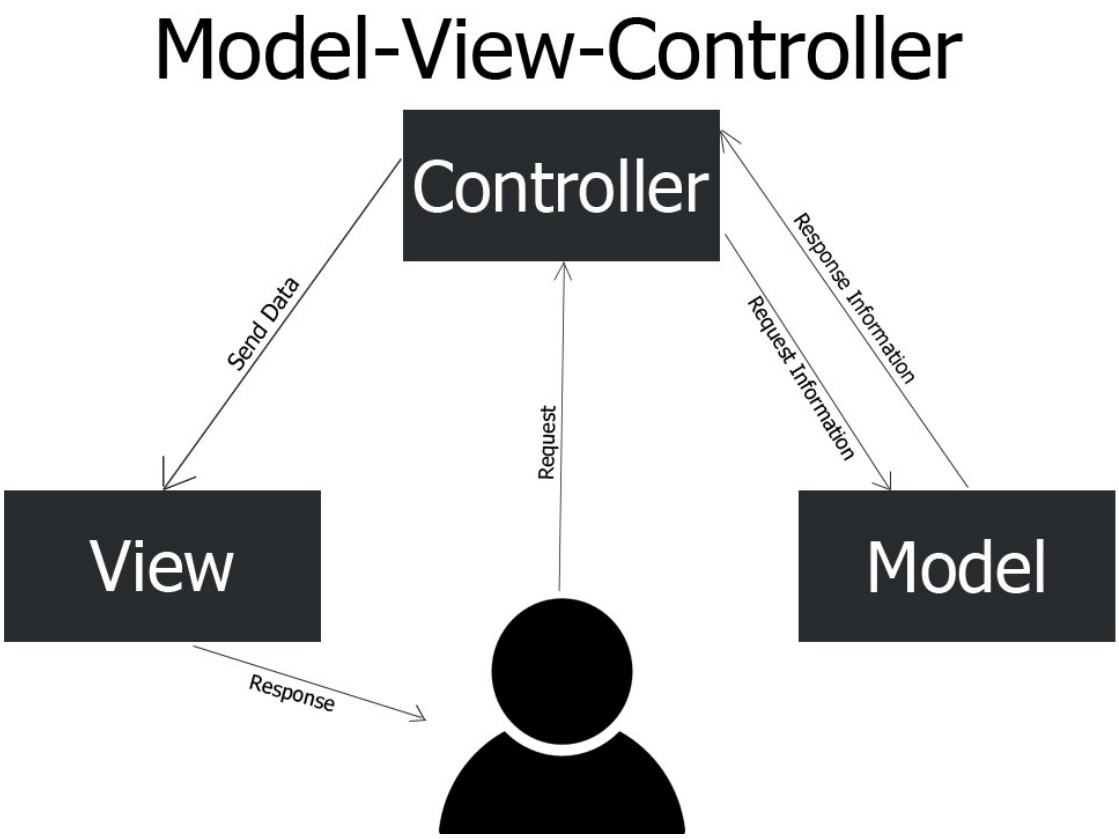
โดยที่สถาปัตยกรรมซอฟต์แวร์ที่ yesod ใช้เป็นรูปแบบ Model-View-Controller (MVC) หนึ่งในเป้าหมาย ของ MVC คือแยกส่วนของ logic ออกจากส่วนติดต่อผู้ใช้ view เพื่อที่จะทำการแยกได้โดยใช้ภาษาเทมเพลต (template language) อย่างไรก็ตาม มีหลายวิธีในการแก้ไขปัญหานี้ ที่ปลายด้านหนึ่งของสเปกตรัม ตัวอย่างเช่น PHP/ASP/JSP จะอนุญาตให้คุณฝังโค้ดใดก็ได้ภายในเทมเพลตของคุณ อีกด้านหนึ่ง คุณมีระบบ เช่น StringTemplate และ QuickSilver ซึ่งผ่าน argument และไม่มีการโต้ตอบกับส่วนที่เหลือของโปรแกรม

แต่ละระบบมีข้อดีและข้อเสีย การมีระบบเทมเพลตที่มีประสิทธิภาพยิ่งขึ้นสามารถอำนวยความสะดวกได้อย่างมาก หากต้องการแสดงเนื้อหาของตารางฐานข้อมูลหรือไม่? ไม่มีปัญหา ดึงเข้าไปด้วยเทมเพลต อย่างไรก็ตาม วิธีการ ดังกล่าวสามารถนำไปสู่โค้ดที่ซับซ้อนได้อย่างรวดเร็ว การอัพเดตเคอร์เซอร์ฐานข้อมูลแบบสลับกันด้วยการสร้าง HTML สิ่งนี้สามารถเห็นได้ทั่วไปในโครงการ ASP ที่เขียนไม่ดี

แม้ว่าระบบเทมเพลตที่อ่อนแอจะสร้างโค้ดง่ายๆ แต่ก็มีแนวโน้มที่จะทำงานซ้ำซาก คุณมักจะต้องไม่เพียงแค่เก็บค่า ดั้งเดิมไว้ในประเภทข้อมูลเท่านั้น แต่ยังต้องสร้าง dictionary ของค่าเพื่อส่งต่อไปยังเทมเพลตด้วย การดูแลรักษา โค้ดดังกล่าวไม่ใช่เรื่องง่าย และโดยปกติแล้วจะไม่มีทางที่คอมไพเลอร์จะช่วยคุณได้

ภาษาเทมเพลตของตระกูล Yesod ภาษาเช็คสเปียร์ (Shakespear) มุ่งมั่นเพื่อความเป็นกลางโดยใช้ประโยชน์ จากความโปร่งใสในการอ้างอิงมาตรฐานของ Haskell เราสามารถมั่นใจได้ว่าเทมเพลตของเราไม่มีผลข้างเคียง อย่างไรก็ตาม พวกเขายังคงสามารถเข้าถึงตัวแปรและฟังก์ชันทั้งหมดที่มีอยู่ในโค้ด Haskell ของคุณได้อย่างเต็มที่ นอกจากนี้ เนื่องจากได้รับการตรวจสอบอย่างครบถ้วนสำหรับทั้งรูปร่างที่ดี ความละเอียดที่เปลี่ยนแปลงได้ และ ความปลอดภัยในการพิมพ์ ณ เวลา compile time การพิมพ์ผิดจึงมีโอกาสน้อยมากที่คุณจะต้องค้นหาโค้ดของ คุณที่พยายามจะปักหมุดจุดบกพร่อง

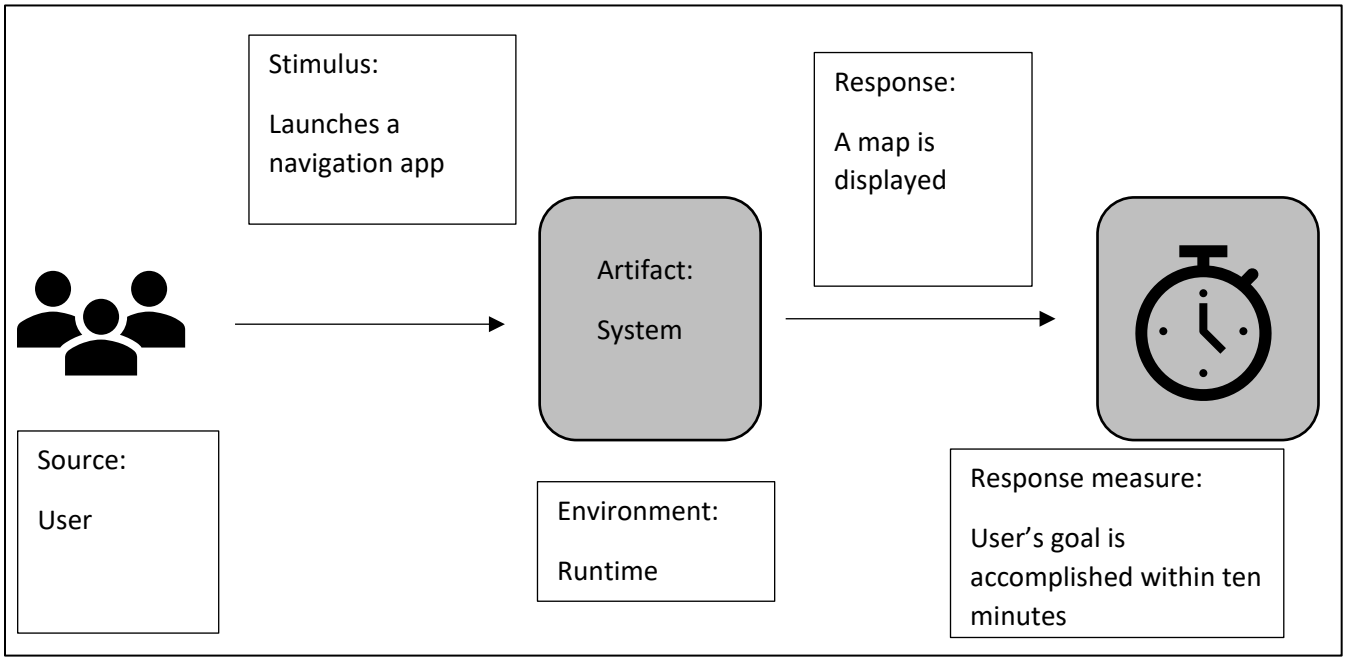
รูปภาประกอบ MVC



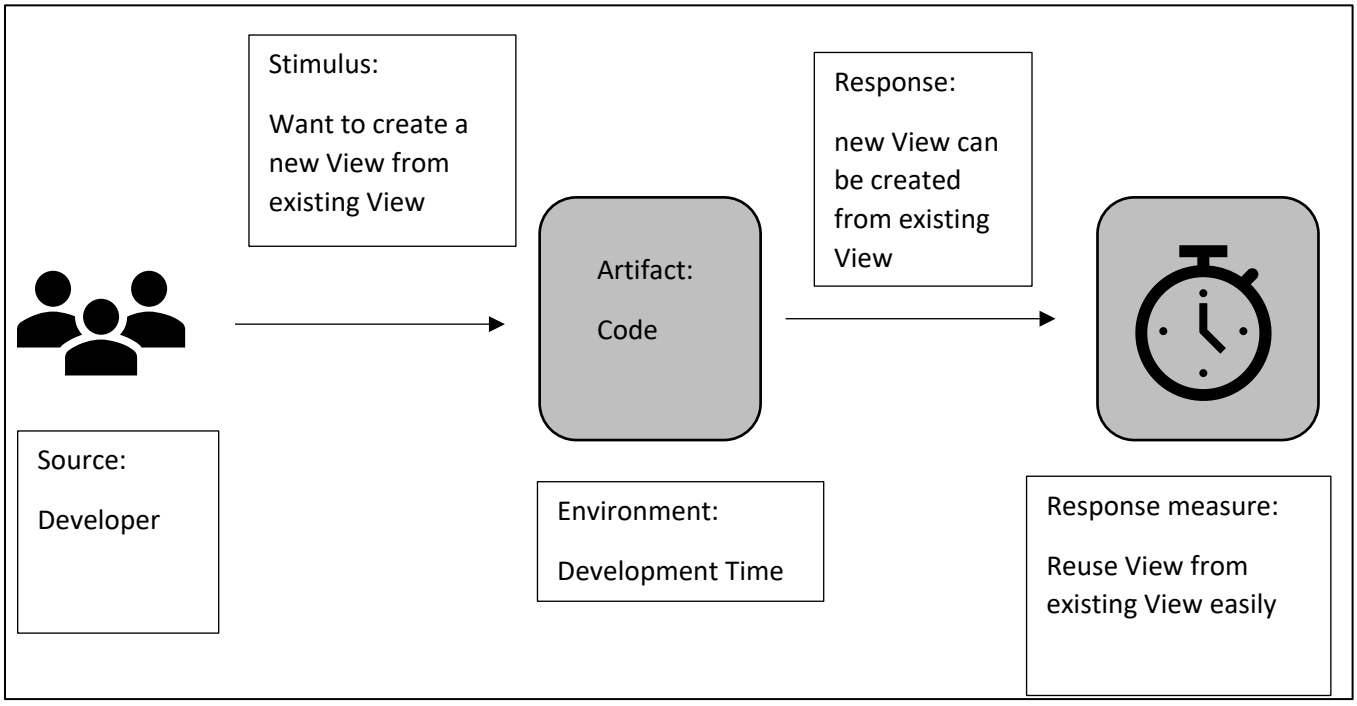
- View แสดงผลค่าในโมเดลในรูปแบบที่เหมาะสมต่อการปฏิสัมพันธ์กับผู้ใช้ ในแต่ละโมเดลสามารถมีวิวได้หลายแบบ เพื่อใช้ในจุดประสงค์ที่ต่างกันโดยใน yesod ใช้ Shakespeare
- Controller รับข้อมูลจากผู้ใช้เข้ามา แล้วดำเนินการตอบสนองต่อข้อมูลนั้น โดยเรียกใช้ logic ต่างๆจากอ็อบเจกต์ในโมเดล และส่งข้อมูลผลลัพธ์นั้นกลับไปยังส่วนแสดงผล เพื่อตอบกลับไปยังผู้ใช้ได้อย่างถูกต้อง
- Model หมายถึง ส่วนของซอฟต์แวร์ที่ใช้ในการแปลการทำงานของระบบ ไปสู่สิ่งที่ระบบซอฟต์แวร์ได้ถูกออกแบบเอาไว้ ตรรกะเนื้อหาใช้เพื่อให้ความหมายแก่ข้อมูลดิบ (ยกตัวอย่างเช่น การคำนวณว่าวันนี้เป็นวันเกิดของผู้ใช้หรือไม่, หรือจำนวนเงินรวม ภาษี และค่าส่งสินค้า ในตะกร้าสินค้า) เมื่อโมเดลมีการเปลี่ยนแปลง จะมีการส่งคำเตือนให้แก่ วิว ที่เกี่ยวข้องเพื่อปรับค่าระบบซอฟต์แวร์หลายระบบใช้การเก็บข้อมูลถาวร เช่น ฐานข้อมูล เพื่อเก็บข้อมูลเหล่านี้ MVC ไม่ได้กำหนดถึงระดับการเข้าถึงข้อมูล เพราะเป็นที่เข้าใจกันว่าส่วนนี้จะอยู่ภายใต้ หรือถูกครอบคลุมด้วยโมเดล โมเดลไม่ได้เป็นเพียงอ็อบเจกต์ที่ใช้เข้าถึงข้อมูล แต่ในระบบซอฟต์แวร์เล็กๆ ซึ่งมีความซับซ้อนน้อยจะไม่เห็นความแตกต่างมากนักโดยใน yesod ใช้ Persistent เป็น model

Quality Attribute Scenarios

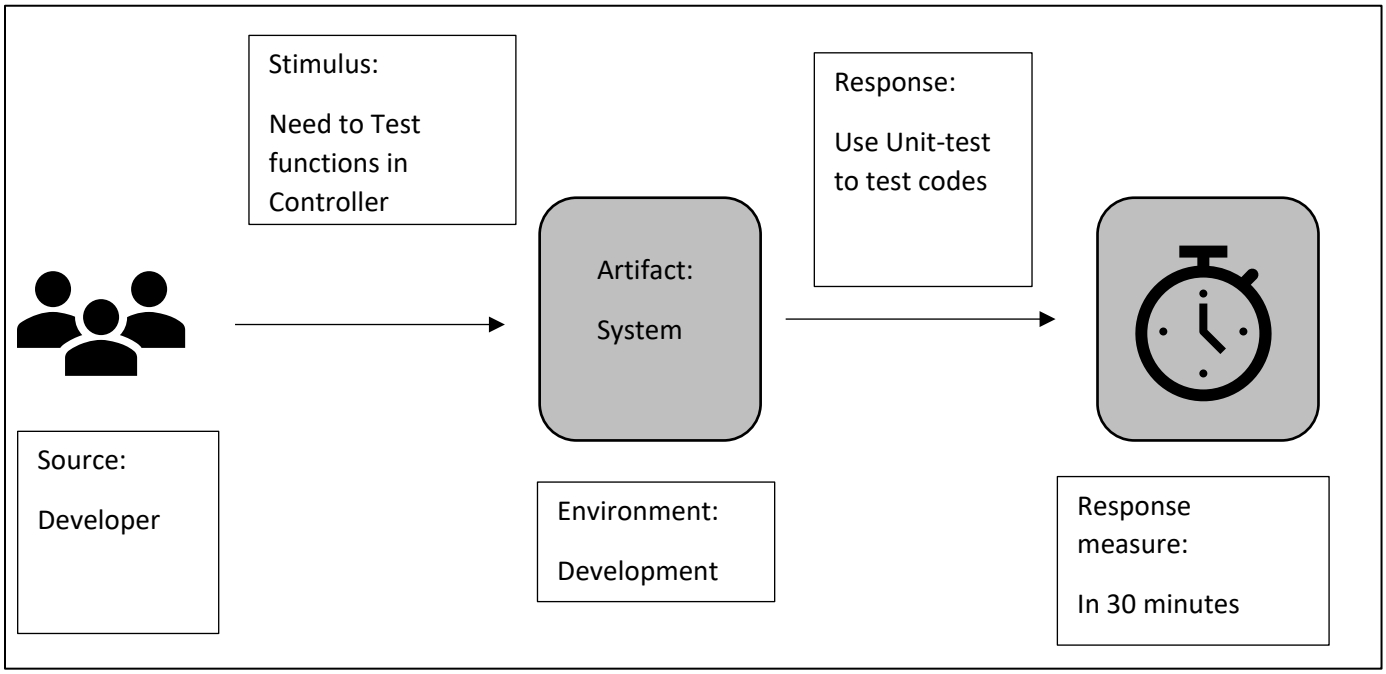
Usability



Reusability



Testability



แหล่งอ้างอิง

- https://www.google.com/search?q=mvc&hl=en&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjPs76q8ov6AhUUiOYKHZW9C-0Q_AUoAXoECAIQAw&biw=1920&bih=947&dpr=1#imgrc=Y9TvT6_QwerjoM
- https://en.wikipedia.org/wiki/Yesod_%28web_framework%29#MVC_architecture
- https://stackoverflow.com/questions/23239236/yesod-architecture
- https://www.aosabook.org/en/yesod.html
- https://www.yesodweb.com/
- https://th.wikipedia.org/wiki/%E0%B9%82%E0%B8%A1%E0%B9%80%E0%B8%94%E0%B8%A5-%E0%B8%A7%E0%B8%B4%E0%B8%A7-%E0%B8%84%E0%B8%AD%E0%B8%99%E0%B9%82%E0%B8%97%E0%B8%A3%E0%B8%A5%E0%B9%80%E0%B8%A5%E0%B8%AD%E0%B8%A3%E0%B9%8C#:~:text=Model%2DView%2DController%20(MVC,%E0%B8%8A%E0%B9%88%E0%B8%A7%E0%B8%A2%E0%B9%83%E0%B8%AB%E0%B9%89%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B8%9E%E0%B8%B1%E0%B8%92%E0%B8%99%E0%B8%B2%20%E0%B8%81%E0%B8%B2%E0%B8%A3