Team Amadeus: MAD Assembly Builder Design Review

Members:

Wyatt Evans, Kyle Krueger,
Melody Pressley, Evan Russell

Mentor:

Austin Sanders

Sponsors:

Dr. Hélène Coullon & Frédéric Loulergue

Team Introductions

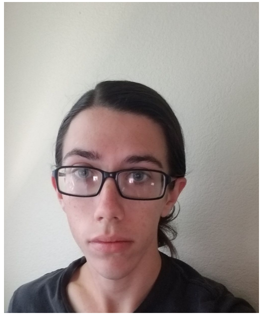
Wyatt Evans

Team Leader

Kyle Krueger

Release Manager

Melody Pressley



Document Architect

Evan Russell

Documenter

Software Deployment

- Deployment of software across multiple devices
- Many interrelated, interconnected activities
- All software is unique
  - Different dependencies, characteristics, specifications
  - Deployment process must be unique

Fig. 1: Software Deployment Example

Our Clients

NORTHERN ARIZONA UNIVERSITY

Dr. Frédéric Loulergue

Professor @ School of Informatics Computing and Cyber Systems

Inria
inventors for the digital world

Dr. Hélène Coullon

Assistant Professor at IMT Atlantique, Inria researcher

Madeus / MAD

- Madeus
  - Theoretical Model for Software Deployment
  - Explicitly Defined Steps and Dependencies
- MAD
  - Madeus Application Deployer
  - Formal Implementation
  - Python

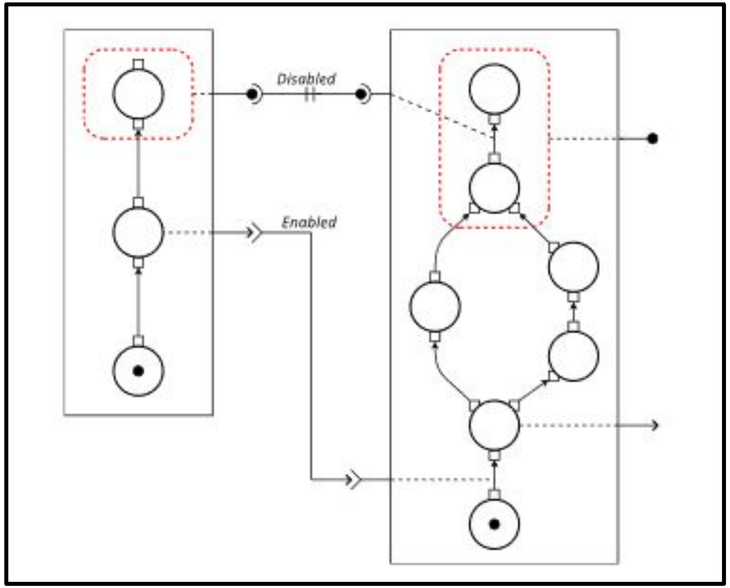


Fig. 2: Basic Madeus Assembly

The Problem

- Current process is slow
- Designing an assembly in code is tedious
- Complex to edit
- Easier to visualize and modify with diagrams

Our Solution: Develop a GUI

- Visualization
- Simulation
- Easy for user to edit
- Decrease turnaround time on MAD Assembly development

Key Requirements

- Visualize & Simulate Madeus Assemblies
- Generate MAD code that represents the user’s diagram
- Extensible Framework that allows for future additions

Top Level Requirements

- Functional
  - Drag-and-Drop method for building Madeus Assemblies
  - Animations & Graphics for Simulation of the Assemblies
  - GUI representation can generate MAD Code
  - Save Assemblies to .yaml files
  - Unobtrusive alert system (deadlocks, incompatible layout, etc.)
  - Plugin Support to allow for forward-thinking extensibility

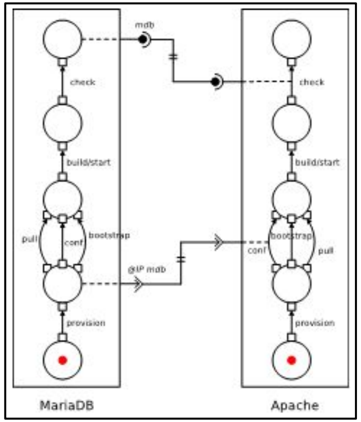
Top Level Requirements (cont.)

- Performance
  - A basic 2-component Assembly with 3 places in each can be built in less than 30 minutes
  - Simulates the Assembly accurately; within 5 seconds of projected time.
  - Saves an Assembly within 1 minute
- Environmental
  - Generated Code is in Python
  - Cross-Platform: Windows, MacOS, Linux

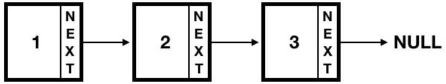
A Breakdown of Code Generation (Process)

- The user creates an assembly and component(s).
- Component creation simultaneously creates a back-end linked list.
- When the user requests code generation, iterate over the linked list to create each component file.
- After each component file has been created, the driver program will be created based off variables in the component programs(s).
- The user can then run the program if needed.

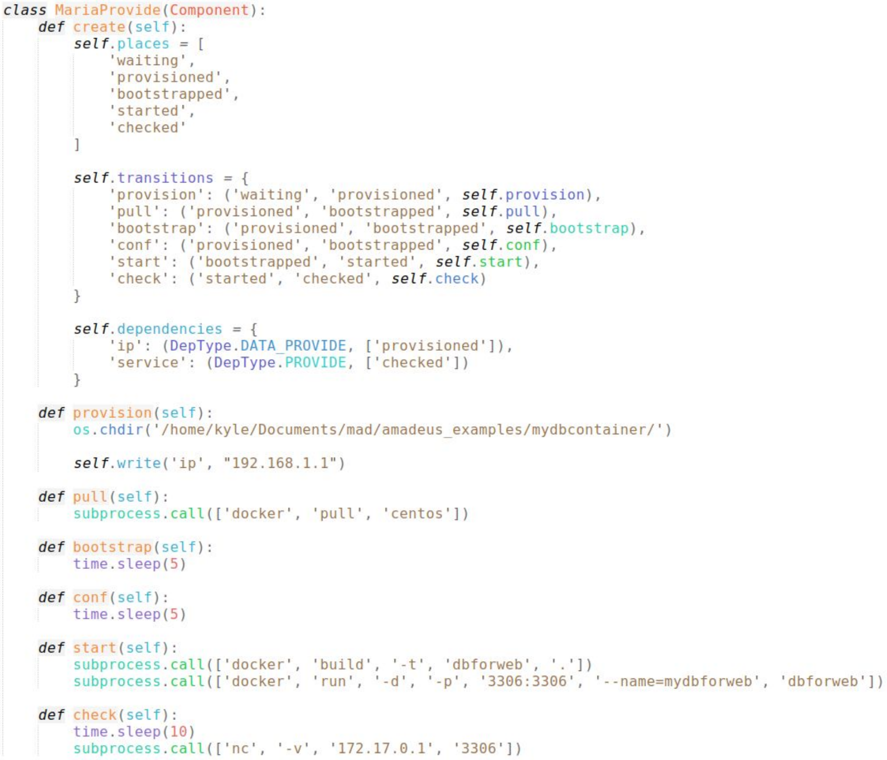
A Graphical Breakdown of Code Generation



(1) - GUI Front-End



(2) - Interface



```
class MariaProvide(Component):
    def create(self):
        self.places = [
            'waiting',
            'provisioned',
            'bootstrapped',
            'started',
            'checked'
        ]

        self.transitions = {
            'provision': ('waiting', 'provisioned', self.provision),
            'pull': ('provisioned', 'bootstrapped', self.pull),
            'bootstrap': ('provisioned', 'bootstrapped', self.bootstrap),
            'conf': ('provisioned', 'bootstrapped', self.conf),
            'start': ('bootstrapped', 'started', self.start),
            'check': ('started', 'checked', self.check)
        }

        self.dependencies = {
            'ip': (DepType.DATA_PROVIDE, ['provisioned']),
            'service': (DepType.PROVIDE, ['checked'])
        }

    def provision(self):
        os.chdir('/home/kyle/Documents/mad/amadeus_examples/mydbcontainer/')

        self.write('ip', "192.168.1.1")

    def pull(self):
        subprocess.call(['docker', 'pull', 'centos'])

    def bootstrap(self):
        time.sleep(5)

    def conf(self):
        time.sleep(5)

    def start(self):
        subprocess.call(['docker', 'build', '-t', 'dbforweb', '.'])
        subprocess.call(['docker', 'run', '-d', '-p', '3306:3306', '--name=mydbforweb', 'dbforweb'])

    def check(self):
        time.sleep(10)
        subprocess.call(['nc', '-v', '172.17.0.1', '3306'])
```

(3) - Generated Code

Overall Requirement Summary

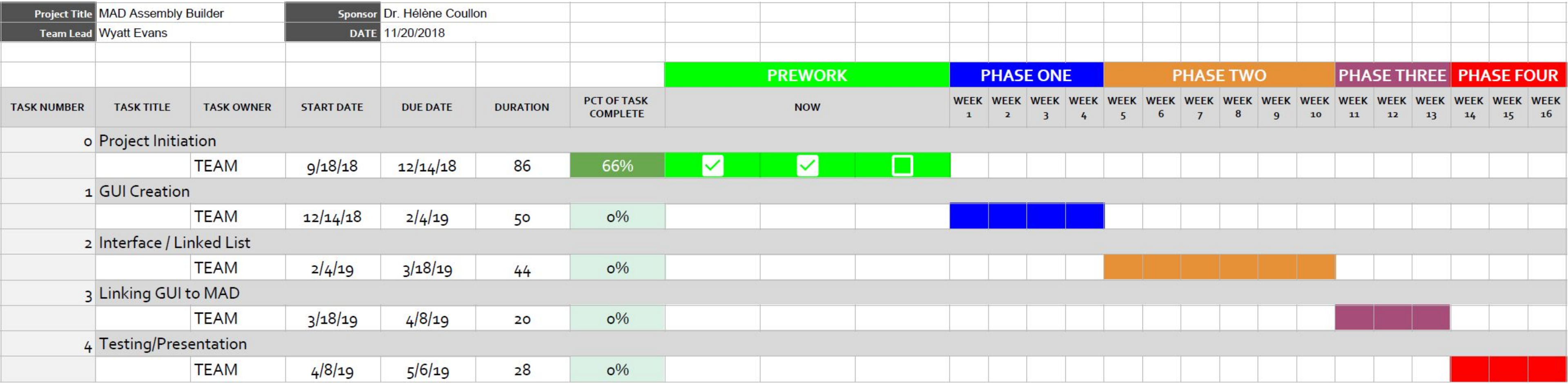
- Code generation
- Real-time animation/simulation in GUI
- Future plug-in support

Risks and Feasibility

- Generated MAD Code may not accurately represent GUI diagram
  - Result in incorrectly deployed software which could lead to infrastructure instability
  - Develop cohesive tests of a simple assembly (MariaDB and Apache)
  - Test edge cases that may also break the back-end MAD code generation
- Software Integrity
  - Ensuring software is extensible with plugins while keeping software integrity
  - Allowing the user to create plugins without altering the code foundation
- Simulation time may be inaccurate
  - Minimize overhead
  - Maximize Performance of the animation

Plan Going Forward

Gantt Chart / Development Schedule



Project Title MAD Assembly Builder **Sponsor** Dr. Hélène Coullon
Team Lead Wyatt Evans **DATE** 11/20/2018

							PREWORK			PHASE ONE				PHASE TWO						PHASE THREE			PHASE FOUR		
TASK NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE	NOW			WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15	WEEK 16
0	Project Initiation																								
		TEAM	9/18/18	12/14/18	86	66%	✓	✓	□																
1	GUI Creation																								
		TEAM	12/14/18	2/4/19	50	0%				🟦	🟦	🟦	🟦												
2	Interface / Linked List																								
		TEAM	2/4/19	3/18/19	44	0%								🟧	🟧	🟧	🟧	🟧	🟧						
3	Linking GUI to MAD																								
		TEAM	3/18/19	4/8/19	20	0%														🟪	🟪	🟪			
4	Testing/Presentation																								
		TEAM	4/8/19	5/6/19	28	0%																	🟥	🟥	🟥

Conclusion

- The Problem
  - MAD software results in good deployment performance but is tedious and complicated to implement
  - Need a way to help visualize software deployments
- Our Solution
  - Develop a Graphical User Interface
    1. Help Visualize an Assembly of components with dependencies
    2. Accurately Simulate Software Deployment via animation
    3. Automate the Generation of Madeus Application Deployer Code
- Our Plan Moving Forward
  - Phase 1: GUI Creation

Thank you!

Any questions?