SYSTEMS ENGINEERING
Research Center

DoDAF Methods for Software Engineers

By
C.W. Perr, Dr. Eric Imsand, Dr. John A. Hamilton, Jr.

Annual SERC Research Review
October 5-6, 2011
University of Maryland
Marriott Inn and Convention Center
Hyattsville, MD
www.sercuarc.org

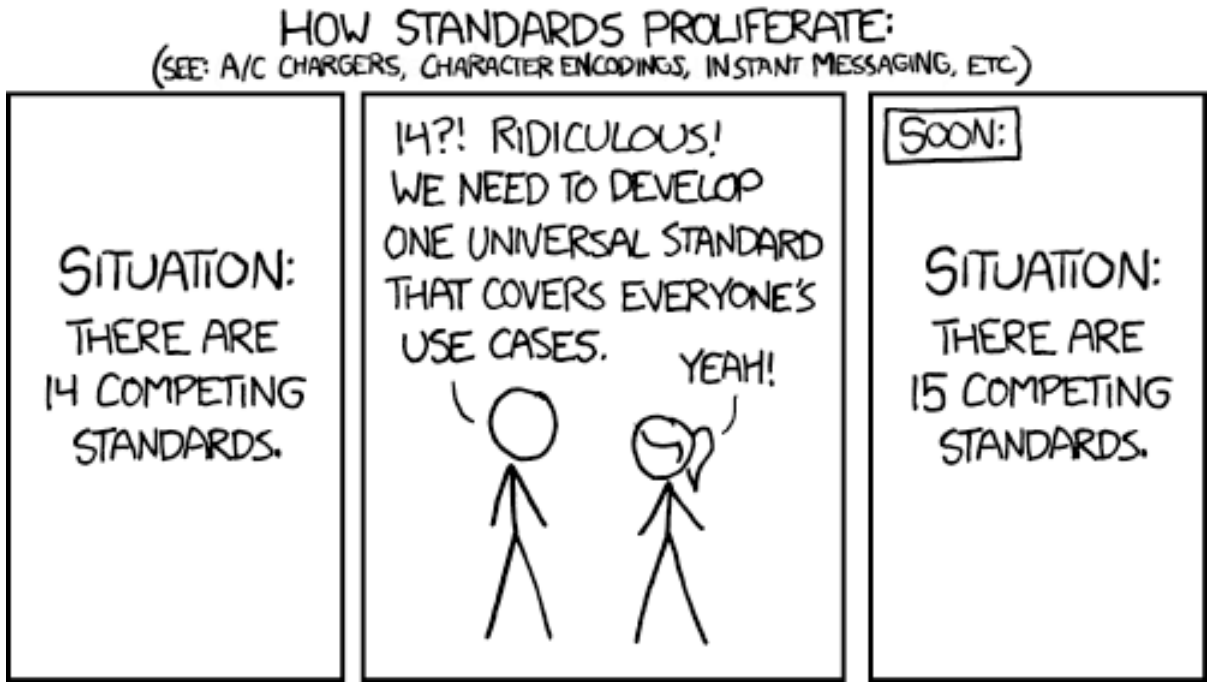
DoDAF

- Designed to reflect architectures of many different types of systems, including software
- Force software engineers to cope with details that might be thought of as extraneous or irrelevant to their work

SYSTEMS ENGINEERING
Research Center

What we are not discussing...

- Is DoDAF the best solution?



The Projects

- RT-24 was looking at how to make DoDAF more useful to software engineers...luckily, we had some software engineers and a sizeable project

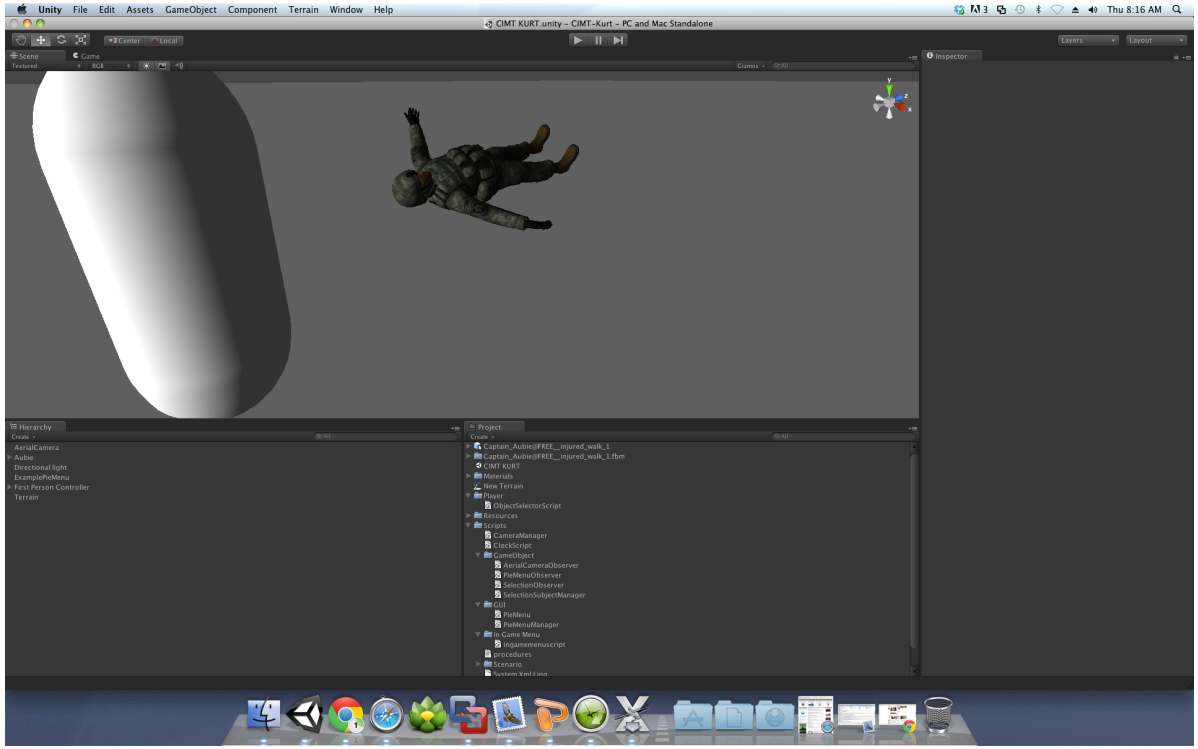
SYSTEMS ENGINEERING
Research Center

C.I.M.T.

- CIMT is the Counter-IED Medical Technology Trainer
- A 3D interactive environment where field medics could train and practice on treating IED victims – also gave new medics additional exposure to the types of graphic injuries they would be seeing

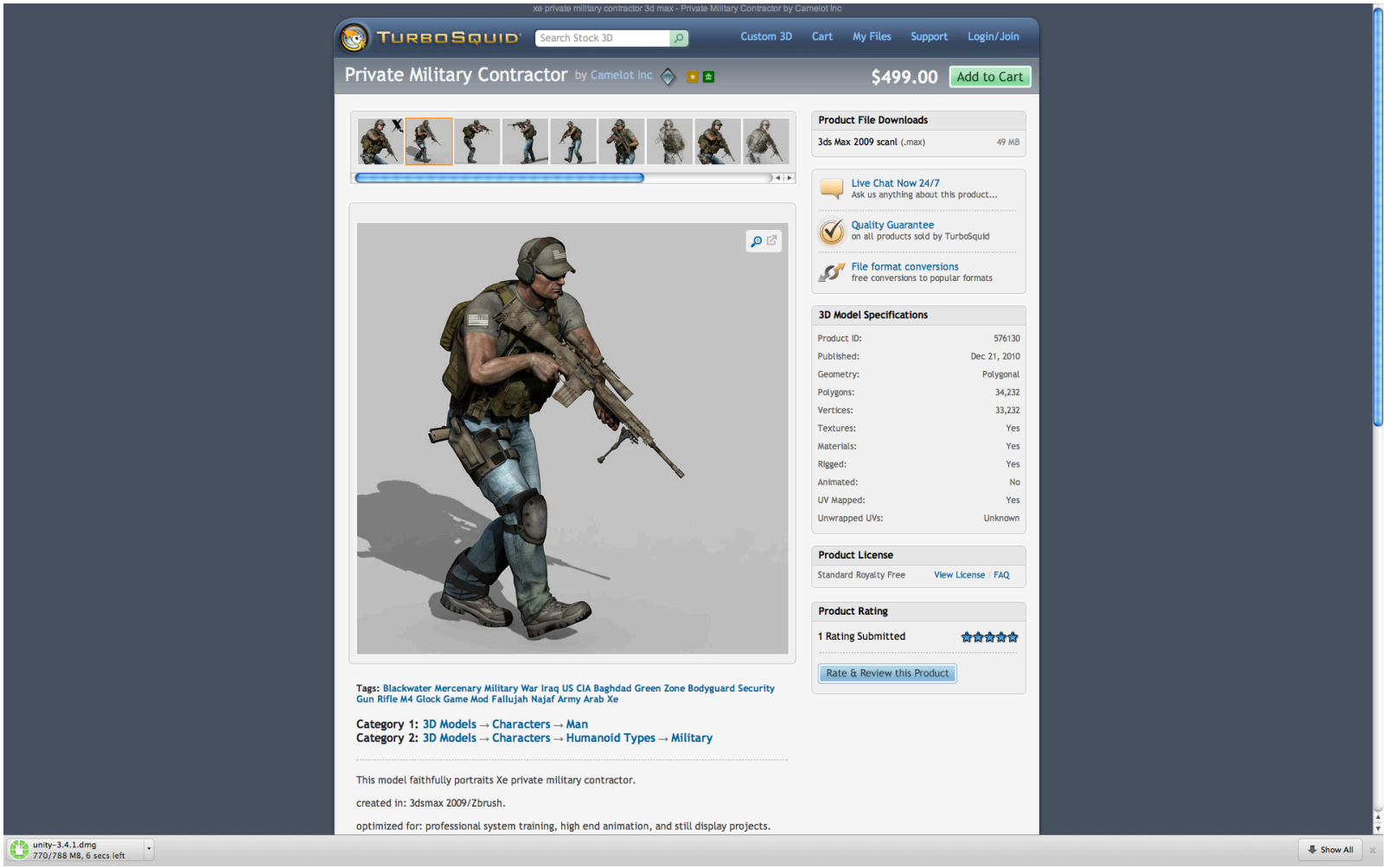
UNITY 3D

- Commercial off-the shelf 3D gaming engine
  - —Good for licensing
  - —Decrease development time
  - —Could use graphics purchased from online retailers



Turbosquid

- Commercial off the shelf graphics



Custom Scripting

```
using UnityEngine;
using System.Collections;

public class ingamemenuscript : MonoBehaviour {
bool inGameMenuToggle = false;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if(Input.GetKeyDown(KeyCode.Escape)){
            if(!inGameMenuToggle){
                inGameMenuToggle = true;
            }
            else{
                inGameMenuToggle = false;
            }
            if(inGameMenuToggle){
            }
        }
    }

    void OnGUI(){
        if(inGameMenuToggle){
            GUILayout.BeginArea(new Rect(Screen.width / 10, Screen.height / 10, 300, 300));
            if(GUILayout.Button("Restart Scenario")){
                Application.LoadLevel("Cleaned Up Scene");
            }
            if(GUILayout.Button("Quit to Main Menu")){
                Application.LoadLevel("TitleMenu");
            }
            if(GUILayout.Button("Exit Application")){
                print("Quitting!");
                Application.Quit();
            }

            GUILayout.EndArea();
        }
    }
}

```

SYSTEMS ENGINEERING
Research Center

Methodology

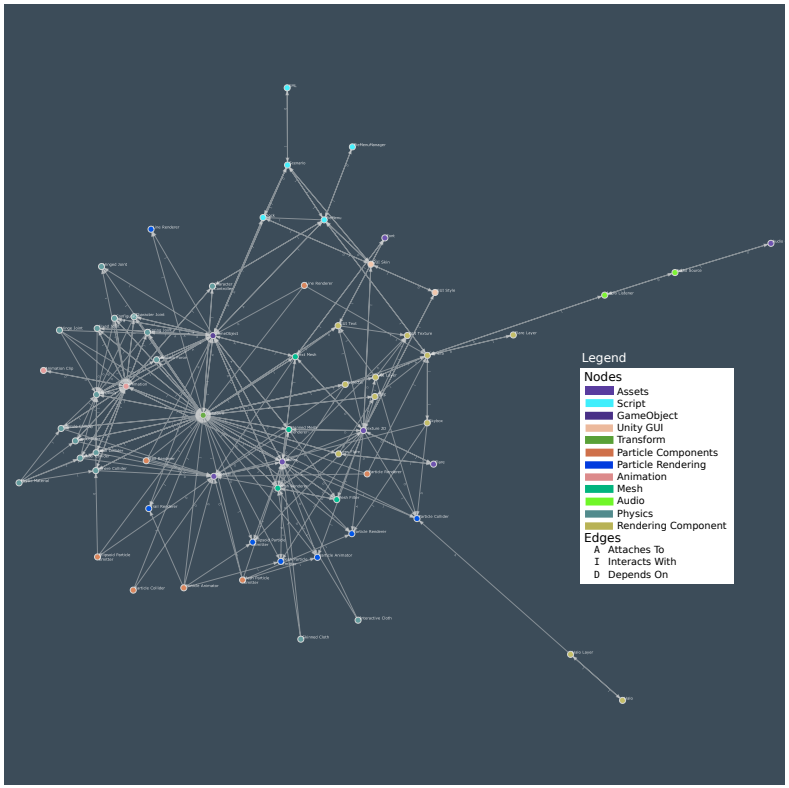
- Modified spiral…weekly spiral development method with a focus on additional features
- Decentralized command with a focus maintained on the developers completing the architecture
- Developers responsibilities closely tracked – progress on the program meant progress/updating done to the architecture

Modifications

- Complaints about having to use DoDAF were common – "why don't we use UML?", "why do I need to know this to make software?", "this isn't value added!"
- The responses: "UML isn't the same, but that will help you", "you do need to understand the problem space…but let's modify where we can", and "nice buzz word usage…but you are wrong"
- The happy medium was found in the decentralized control. You had to do the work, but you could pick your poison. The DoDAF guides were used as references, but not step by step solutions.

The Good

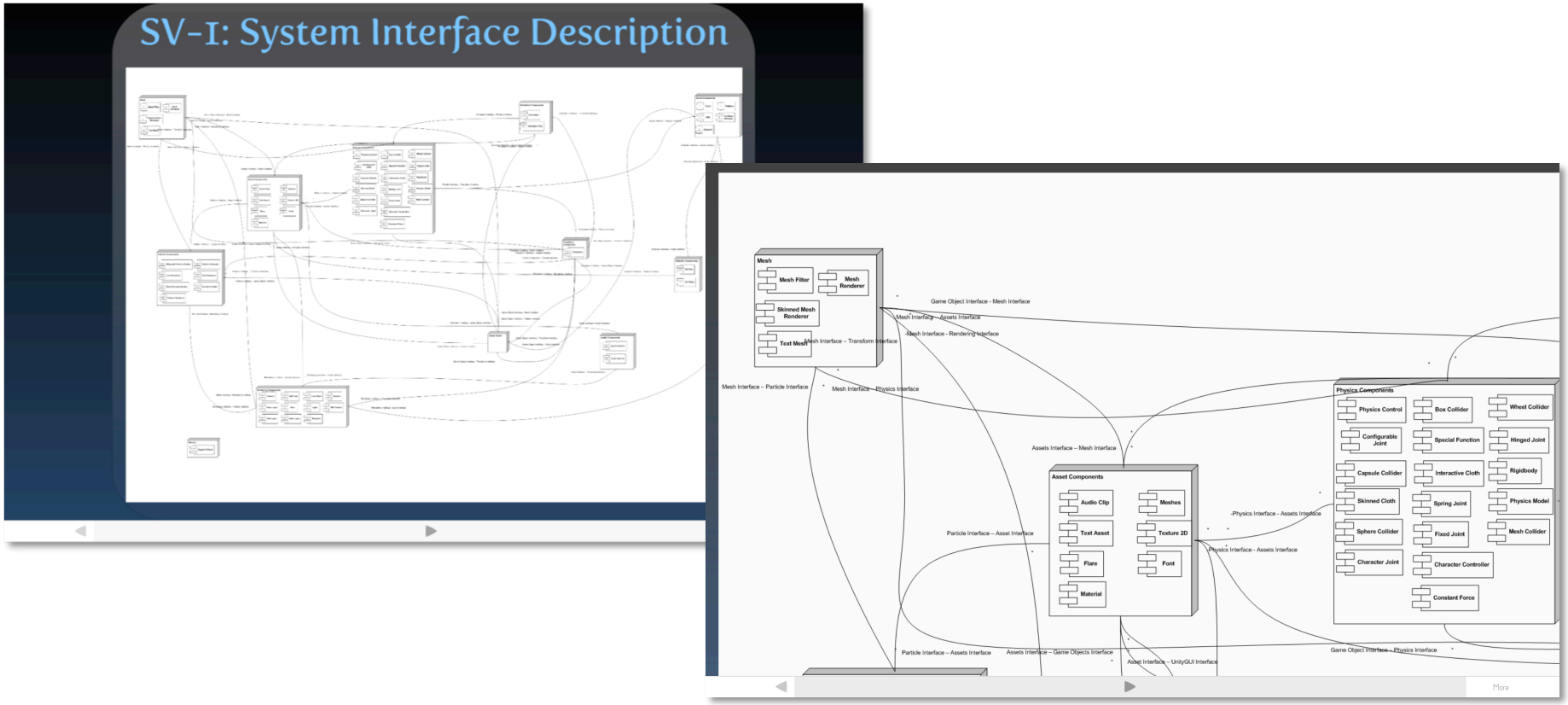
- This led our developers to create methods for automating their work, which usually gave us easier to read and higher quality viewpoints



SYSTEMS ENGINEERING
Research Center

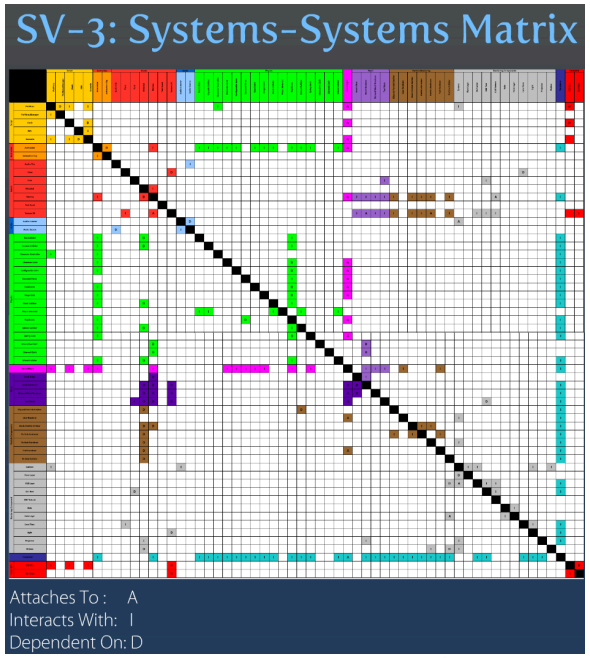
The Good (cont.)

- Prezi - http://prezi.com/xr4rfsfovpv-/dodaf-for-counter-ied-medical-technology/



The Bad

- The developers were able to make convincing arguments that some views should be left out entirely
  - —OV-5a, OV-6a
  - —Looking back, we might have cut more, but instead we used some views to allow for varying level of abstraction
    - Example – SV-2 and SV-3 were closely related, but in using automation we could graphically depict SV-3 inside SV-2



The Ugly

- Before automation, the workload was incredibly high and was taking people away from active work on the software
- Lack of understanding of Unity 3D really slowed down our development
- The main hardship - High learning curve for DoDAF…a two day long DoDAF class was given to get us up to speed

SYSTEMS ENGINEERING
Research Center

Methods/Lessons Learned

- Tell developers the what and not the how
- Encourage automation as long as it leads to quality output
- Bad input = bad output…learn the tool
- Maintain responsibility for developers
- Some LEAN tools apply here…some specialization for the developers, but a need to understand the whole project
  - —In this way anyone should be able to work on the architecture
  - —Creates some backup in the team members (big tests, trips, other issues)

SYSTEMS ENGINEERING
Research Center

Methods/Lessons Learned

- Use the architecture for quality control and verification
  - —Hold the developers to this practice
    - "Does the architecture accurately reflect what you are doing?"
- Towards the middle development on new products where you need a better understanding of the tools

SYSTEMS ENGINEERING
Research Center

Future Work

- Meta-language included in the comments to allow for automatic DoDAF generation (somewhat like IBM's Rational for Rhapsody but with non-strongly formed languages)
- Further refinement of DoDAF work in conjunction with large projects
- Further automation techniques