

UNCLASSIFIED

Raytheon

Customer Success Is Our Mission



Government Open Source Software

GSAW 2013

G. Todd Kaiser
Engineering Fellow

Raytheon IIS

gtkaiser@raytheon.com

303.344.6915

Copyright © 2013 Raytheon Company. All rights reserved.

Published by The Aerospace Corporation with permission. *Customer Success Is Our Mission* is a registered trademark of Raytheon Company.

Copyright © 2013 Raytheon Company. All rights reserved.

NON-ITAR

UNCLASSIFIED

What is “Open Source” software?

Free Software Foundation

Free software is software that gives you the user the freedom to share, study and modify it. We call this free software because the user is free.

"Free Software Foundation." *Free Software Foundation*. N.p., n.d. Web. 27 Jan. 2013. <<http://www.fsf.org/about/>>.

Open Source Initiative

The Open Source Definition [Abridged]

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

- 1. Free Redistribution**
- 2. Source Code**
- 3. Derived Works**
- 4. Integrity of The Author's Source Code**
- 5. No Discrimination Against Persons or Groups**
- 6. No Discrimination Against Fields of Endeavor**
- 7. Distribution of License**
- 8. License Must Not Be Specific to a Product**
- 9. License Must Not Restrict Other Software**
- 10. License Must Be Technology-Neutral**

"The Open Source Definition." *The Open Source Initiative*. N.p., n.d. Web. 27 Jan. 2013. <<http://opensource.org/osd>>.

Open source software ≠ no cost

Government Open Source Software (GOSS)

Open source projects sponsored by government

- Sensitive project access limited to government and approved entities
- Non-sensitive are full-fledged projects on the open internet

Software code, documentation, etc. available to everyone

- Government, contractors, researchers with appropriate access

Community encourages participation

- report defects, share ideas, contribute to the project transparently

Virtual community hosted in government “cloud”

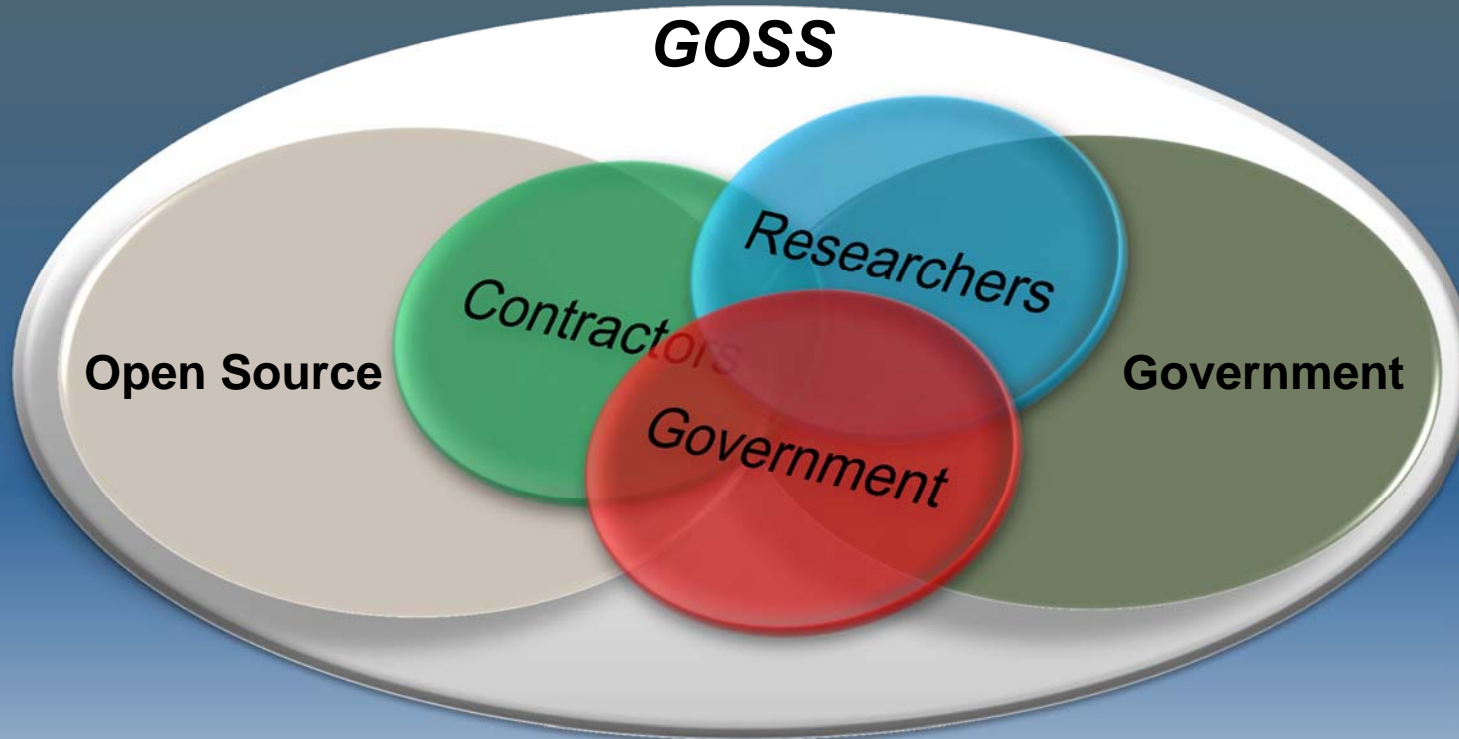
- supports distributed developers

Single entity designated to manage a project

- provides web site, configuration control, forum management, discrepancy tracking, architectural integrity

GOSS follows OSS model

GOSS community



GOSS creates a partnership

GOSS saves money & improves quality

Projects benefits on work done by others

- New projects start with a mature baseline
- Projects benefit from new functionality added by others

Maintenance of a single baseline is cheaper than that for multiple baselines

- Defects fixed by one project benefit all projects
- Single baseline reduces the total lines of software code maintained

Program start up costs are dramatically reduced

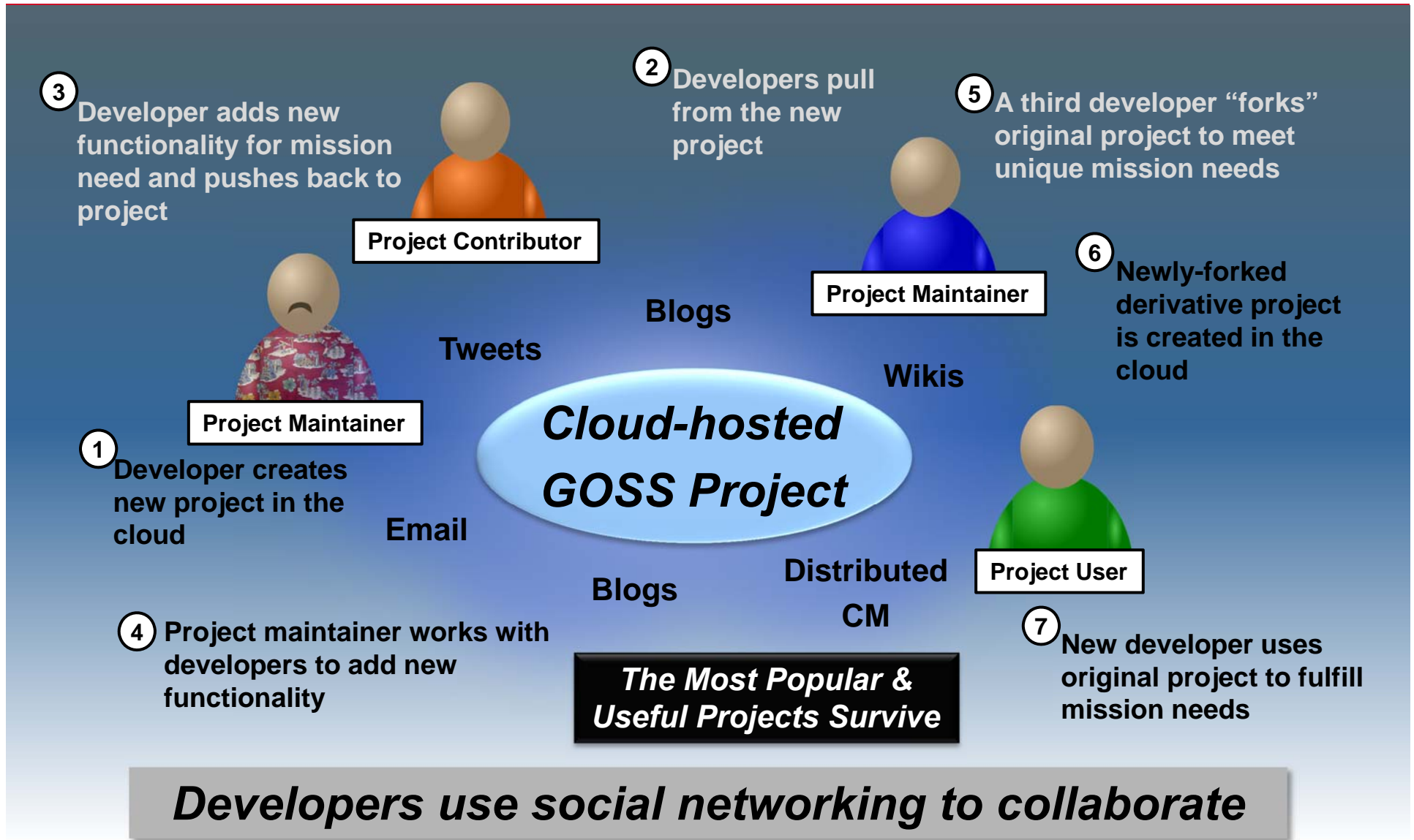
- GOSS baseline is designed and functionality is well known
- Requirements documented by GOSS can be reused on programs

“Darwinism” applies

- Only the most useful projects survive

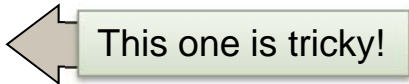
GOSS saves money by reducing duplication

How GOSS/OSS works



Current open source business model

Companies (such as RedHat) make money by:

- Charging for support of their open source-based product
- Charging for value-added functionality 
- Charging for services contracts to install and modify open source-based products to fit unique client needs

Software licenses applied to the open source ensure contributions back to the baseline

- Companies make money on making changes/fixing software, but they have to contribute back
- Companies can market the open source product, but they can't sell licenses to it – only services

Viable business model exists, culture must support

Candidate ground system projects*

- Command and telemetry
- Planning
- Flight Dynamics
- Antenna Control
- Telemetry History
- Calibration Tools
- Messaging
- Payload Processing
- Analysis Tools
- Simulation
- Processor Modeling
- Infrastructure

** - This is NOT to say that a “shared” project for any of these items does not exist today. I do suggest that this is a new model that has rarely been tried in ground systems.*

Many projects that would save time & money

Current defense culture doesn't work

Business climate does not strongly support GOSS

- Contractors building the same software over and over helps sales
- Government budget process does not support sharing software
- No mechanism exists to force project to not "cut and run"

PMs and contractors are inclined to control everything

- Schedule pressure is often used as an excuse to "cut and run"
- My mission is more important than yours
- Cost overruns prevent modifications from being contributed back

Contributing community is potentially limited

- Classified baselines have a smaller contributing community
- Unclassified baselines are often kept classified
- Specialized baselines don't engender many GOSS users

There are no incentives to support GOSS

Making GOSS viable

Provide incentives for GOSS to work:

- Financial incentives for contractors who use and contribute to GOSS projects
- Contractual incentives to ensure that contractors contribute changes back to the appropriate GOSS baseline
- Budget structures to support GOSS baseline and projects who contribute

Provide a mechanism for collaboration

- Servers, networks and storage to host GOSS projects
- Open culture to help PMs and contractors find and participate
- Backing for PMs who push GOSS usage

Include the use of GOSS in proposal scoring criteria

Contractors will use GOSS if they benefit from it

Summary

We've seen an overview of:

- What GOSS is
- The potential for cost savings GOSS can provide
- What the challenges are faced making GOSS viable

What needs to be done:

- We need financial, contractual and maybe legislative incentives for both program offices and contractors
- A collaboration environment needs to be made available to the community

We can make GOSS work

Speaker Bio

Todd Kaiser is an engineer with Raytheon Intelligence and Information Systems (IIS). Todd currently serves as Technical Director for the Enterprise Information Systems (EIS) Command, Control and Communications area of IIS. Todd has a background in software development, satellite command and control and space vehicle dynamics. Todd holds a Master's degree in Computer Science and a Bachelor's degree in physics and mathematics, both from the University of Denver.