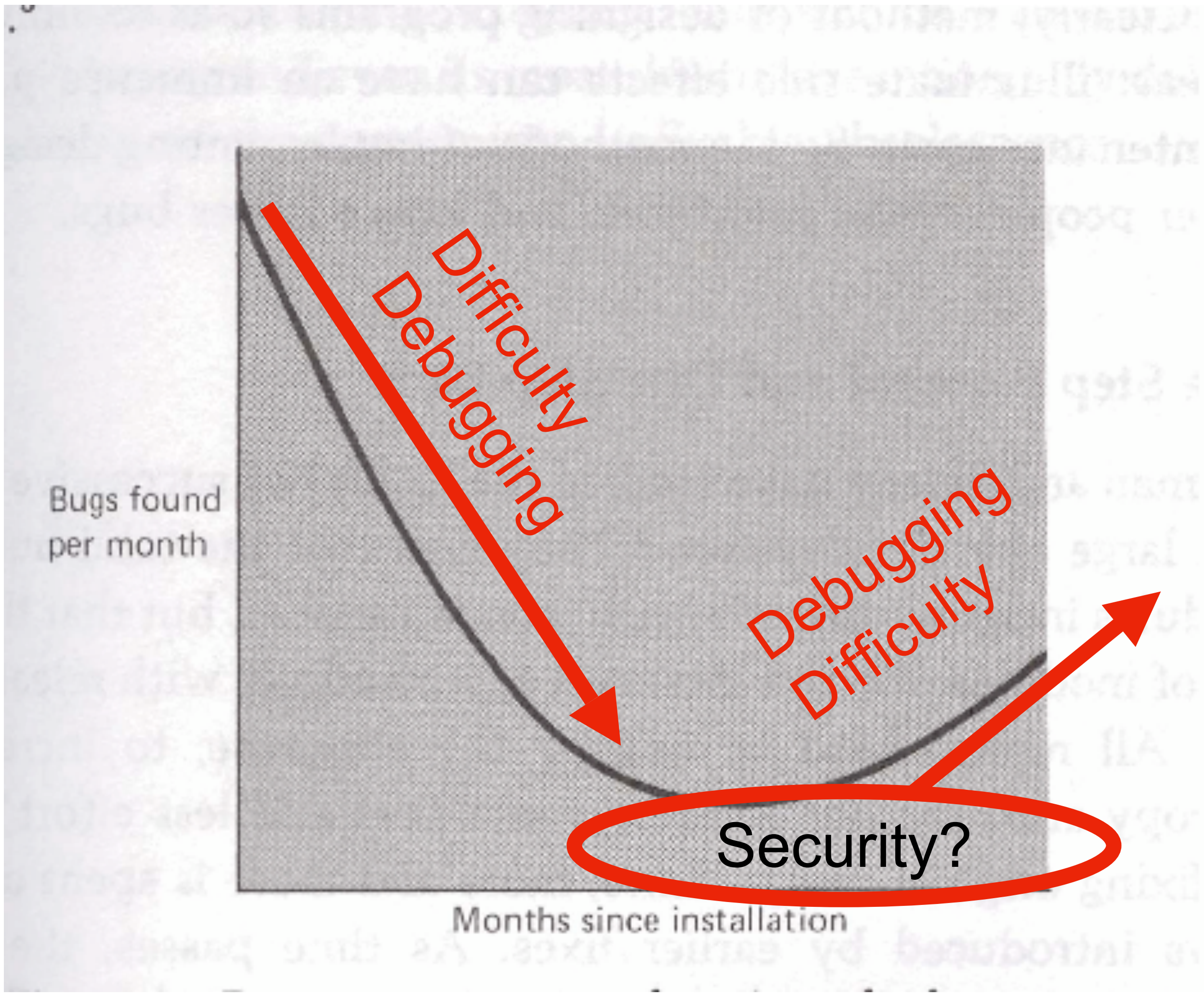
Does Software Quality Matter?

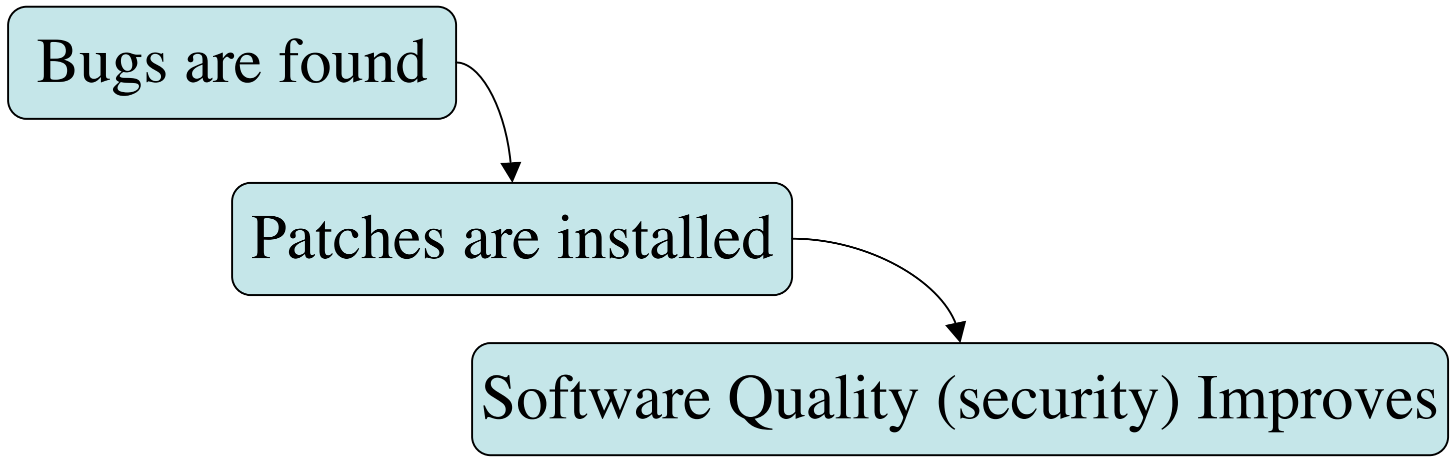
Sandy Clark, Matt Blaze, Jonathan Smith
University of Pennsylvania

Usenix Security: Metricon 2009

Bugs versus time
Mythical Man Month



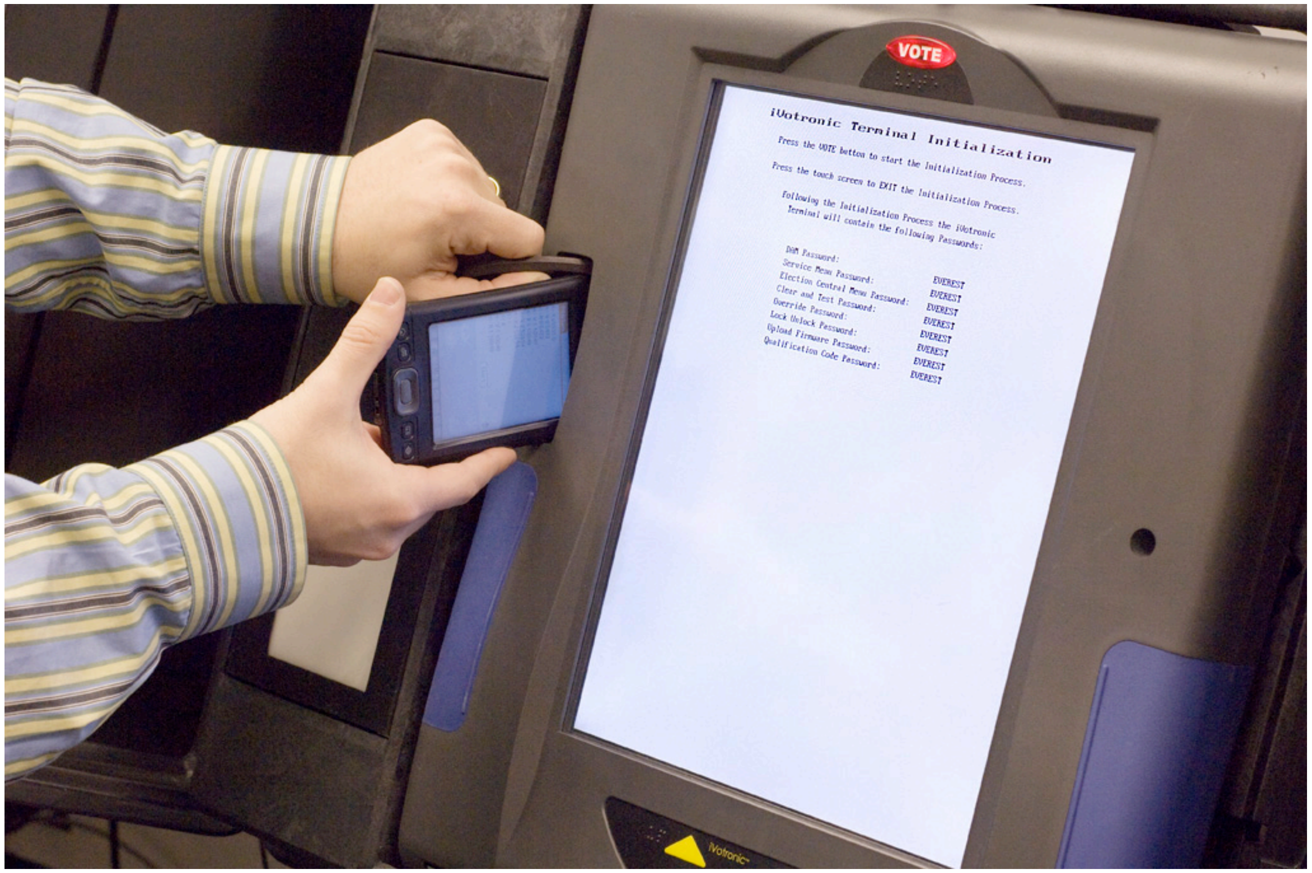
Current Software Engineering Models



The academic community focuses on measuring quantity of vulnerabilities of software

More bugs = More Attacks

A Puzzle for you



```
iVotronic Terminal Initialization

Press the VOTE button to start the Initialization Process.

Press the touch screen to EXIT the Initialization Process.

Following the Initialization Process the iVotronic
Terminal will contain the following Passwords:

DAM Password:
Service Menu Password:                EVEREST
Election Central Menu Password:       EVEREST
Clear and Test Password:              EVEREST
Override Password:                    EVEREST
Lock Unlock Password:                 EVEREST
Upload Firmware Password:             EVEREST
Qualification Code Password:          EVEREST
                                      EVEREST
```

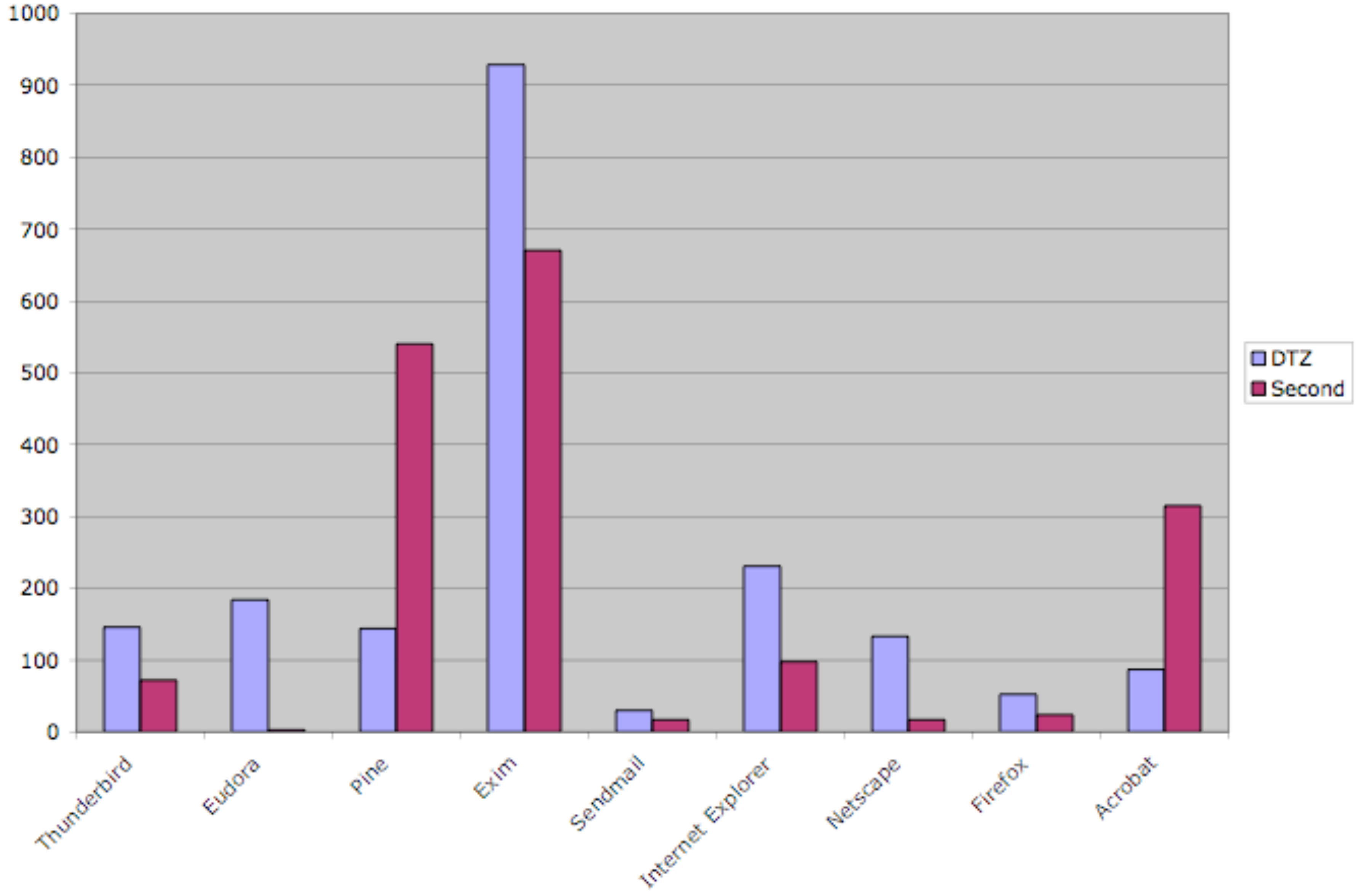
Even weak software enjoys a

Honeymoon

A Honeymoon?

We are defining the Software Honeymoon as the period of time
Between the first release of a program and the disclosure of it’s first exploitable vulnerability.

IOW, **Honeymoon = Learning curve**


1000
900
800
700
600
500
400
300
200
100
0
Thunderbird
Eudora
Pine
Exim
Sendmail
Internet Explorer
Netscape
Firefox
Acrobat
DTZ
Second

Trying to secure software is an Arms Race

The attacker’s curve

We are only beginning to think about individual pieces in the steady state arms race.

- -Papers demonstrating how attackers respond to countermeasures
- -Papers measuring the rates of infection
- -Every “Patch Tuesday” is followed by an “Exploit Wednesday”

But We know almost nothing about the pre-zero day cold war

- Very little research into this area
- We observe that the Honeymoon period is a time of relative peace

An Observation

The Honeymoon ends after attackers:

- Research to discover
  - Exploitable attack vectors
  - Existence of vulnerabilities
    - Soundness of crypto, protocols, implementation, testing
    - **Complexity is attacker's *friend* here**
  - Availability of specs, source code, sample targets
  - Difficulty of finding the vulnerabilities
    - **Complexity is actually the *enemy* of the attacker here**
- Development
  - Build and debug an exploit
- Operations
  - Find and exploit targets

Security Metrics questions

Our usual question: *"Can we measure how secure this system is?"*

We analyze the *intrinsic* properties of the system

A different question: *"Can we measure how long will it be before this system is first attacked?"*

"What is the expected time to zero-day exploit"

We must model not only the intrinsic security of the system, but the threat and behavior of attackers

IOW, *extrinsic* properties are at least as important as *intrinsic* properties

Why this matters?

Intrinsic security properties of software are a poor predictor of when an attack will occur

Intrinsic security properties of software are a poor indicator of how devastating an attack will be

Focus on intrinsic security properties leaves us defenseless against new, innovative attacks

We are spending our attention and resources on the wrong things.

Focus needs to be on extending the Honeymoon

The Arms Race Today

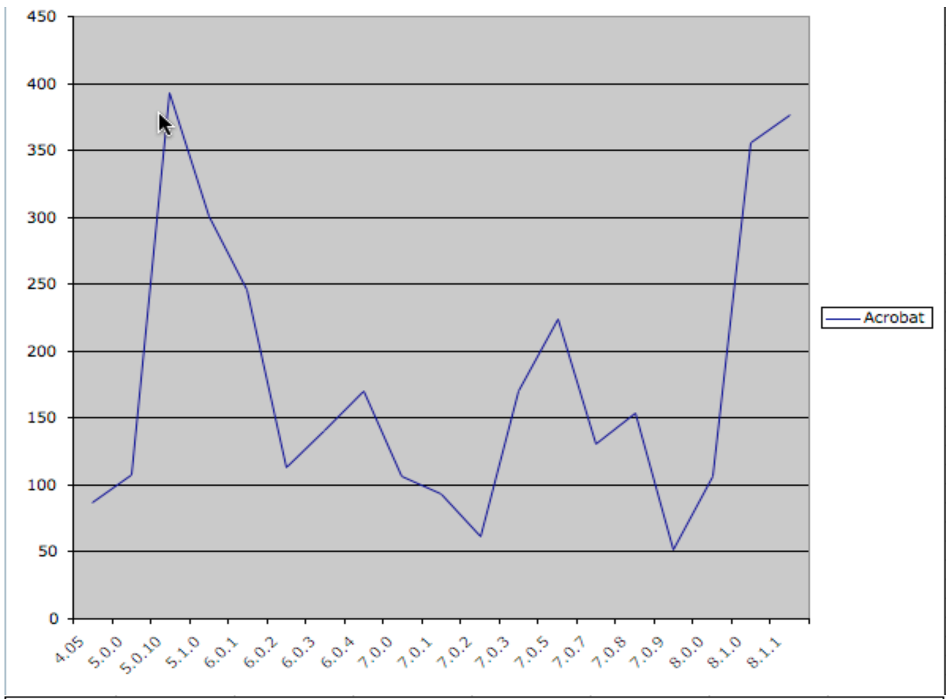
Improving the Extrinsic Properties of Software

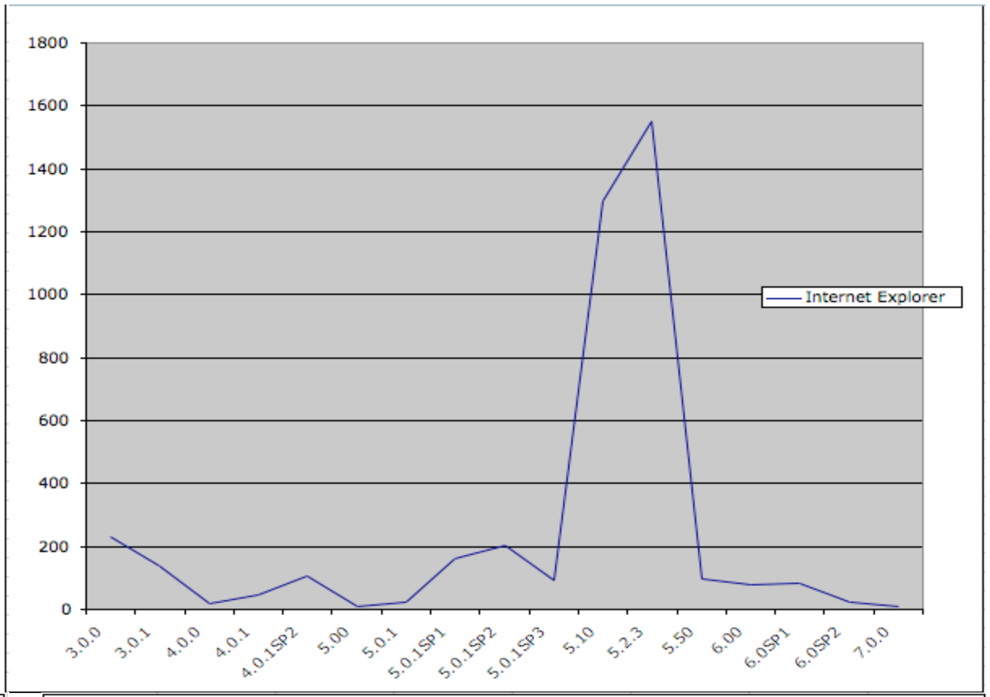
Get a better idea of the length of the Honeymoon period

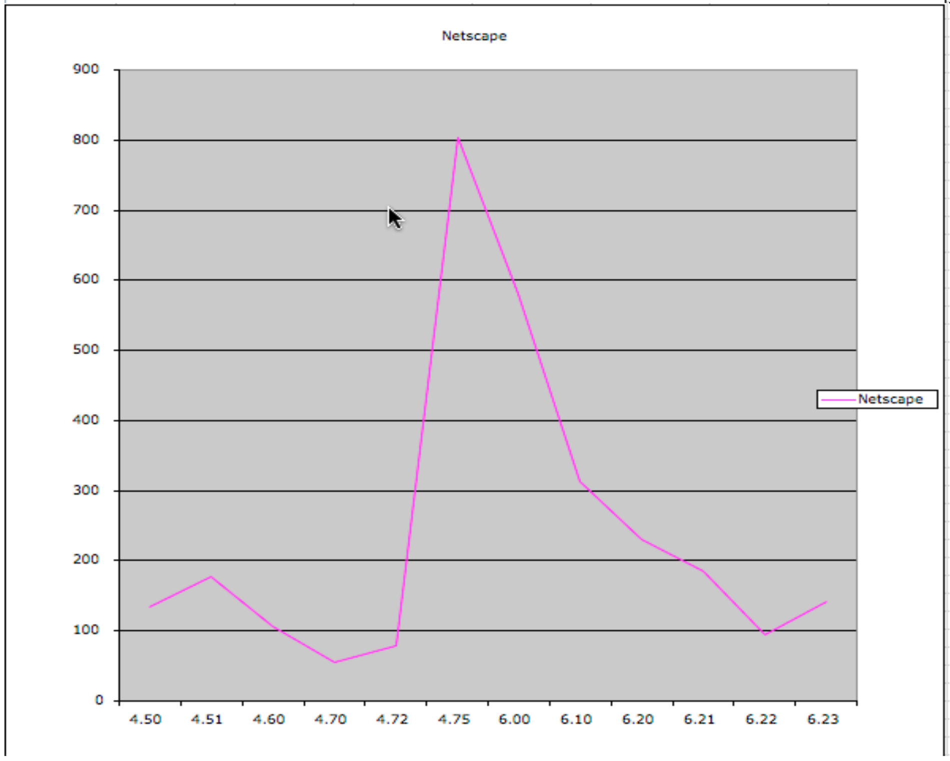
Develop ways to prolong it

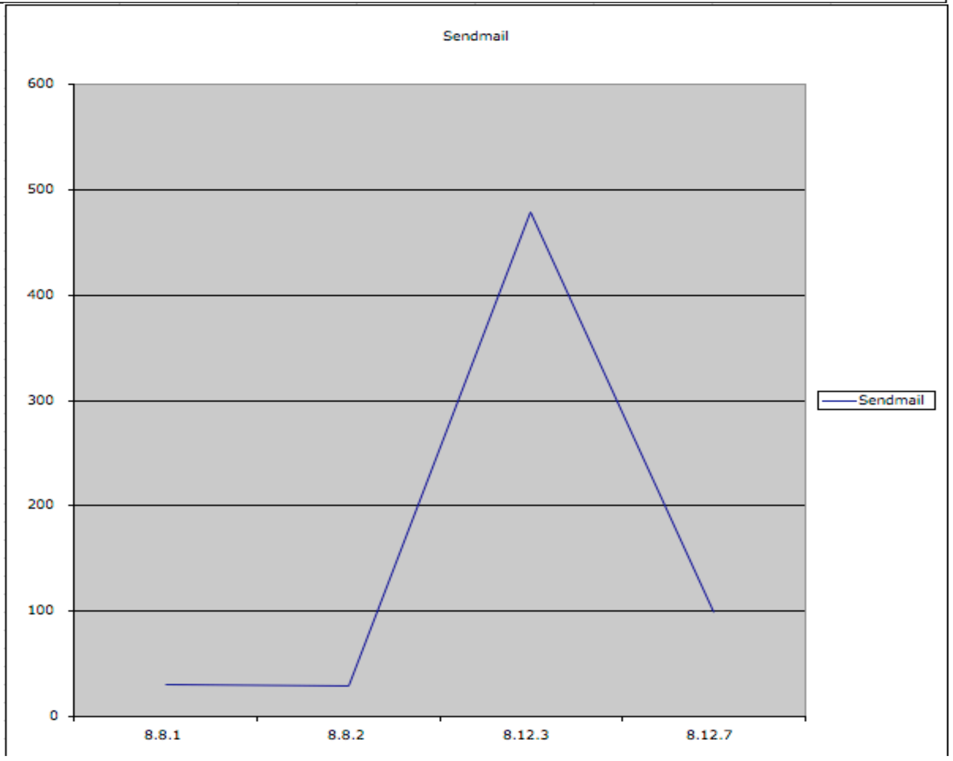
If your honeymoon is over, find ways to jump start it.

May mean completely changing our view of the software life cycle - why?


Acrobat
0
50
100
150
200
250
300
350
400
450
4.05
5.0.0
5.0.10
5.1.0
6.0.1
6.0.2
6.0.3
6.0.4
7.0.0
7.0.1
7.0.2
7.0.3
7.0.5
7.0.7
7.0.8
7.0.9
8.0.0
8.1.0
8.1.1


Internet Explorer
0
200
400
600
800
1000
1200
1400
1600
1800
3.0.0
3.0.1
4.0.0
4.0.1
4.0.1SP2
5.00
5.0.1
5.0.1SP1
5.0.1SP2
5.0.1SP3
5.10
5.2.3
5.50
6.00
6.0SP1
6.0SP2
7.0.0


Netscape
Netscape
0
100
200
300
400
500
600
700
800
900
4.50
4.51
4.60
4.70
4.72
4.75
6.00
6.10
6.20
6.21
6.22
6.23


Sendmail
Sendmail
0
100
200
300
400
500
600
8.8.1
8.8.2
8.12.3
8.12.7

New code is better than old code,
even if it introduces
new vulnerabilities

Because, then you get a second
Honeymoon!