Object-Oriented Software Design
and Software Processes

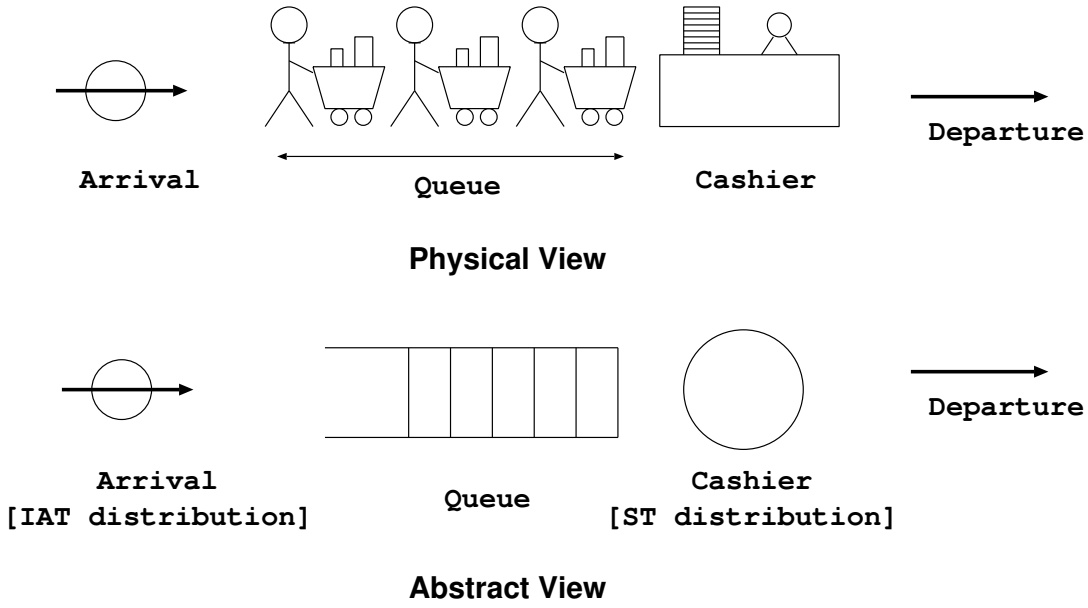
Hans Vangheluwe

Modelling, Simulation and Design Lab (MSDL)
School of Computer Science, McGill University, Montréal, Canada

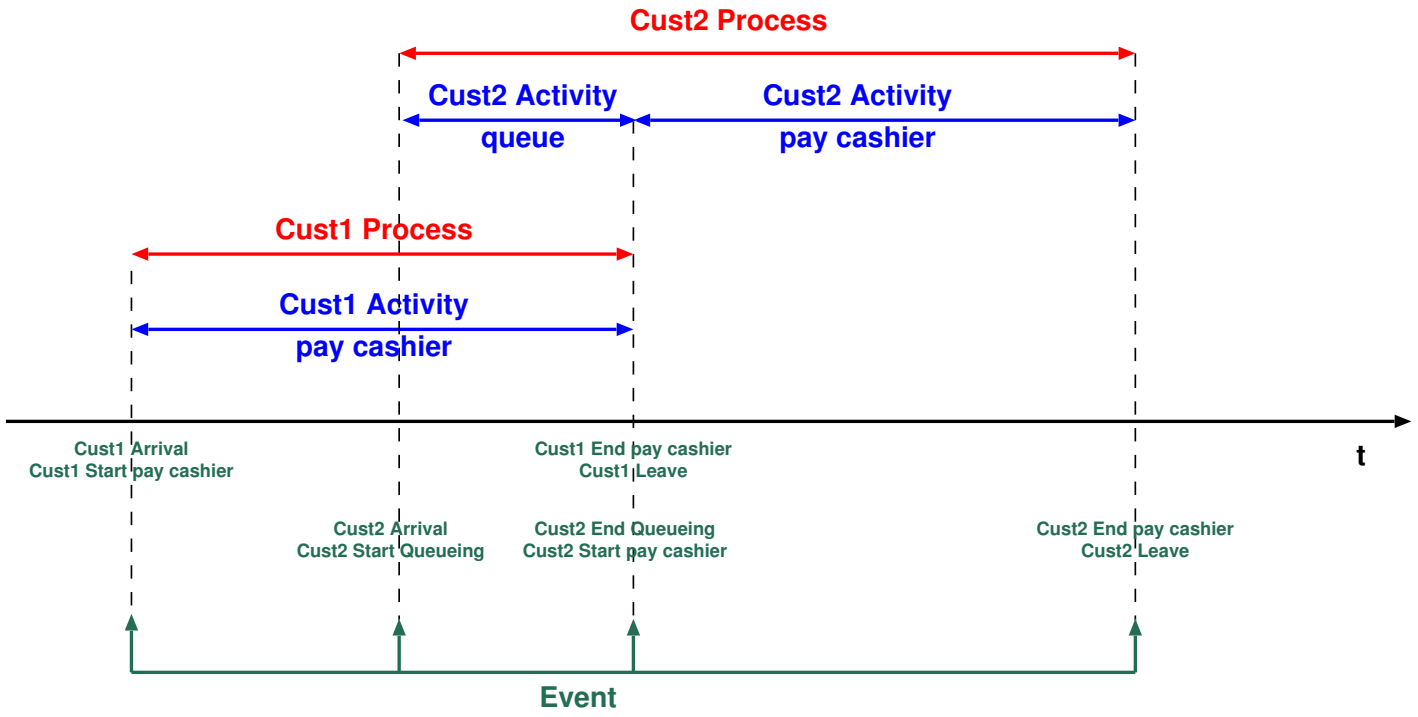
Overview

1. (Software) Process: definition
2. Various Software Processes
3. The Process Influences Productivity:
   Dynamic Process Modelling using Forrester System Dynamics

Process: A Queueing System



Event/Activity/Process



Software Processes

"The Software Engineering **process** is the total set of Software Engineering **activities** needed to transform requirements into software".

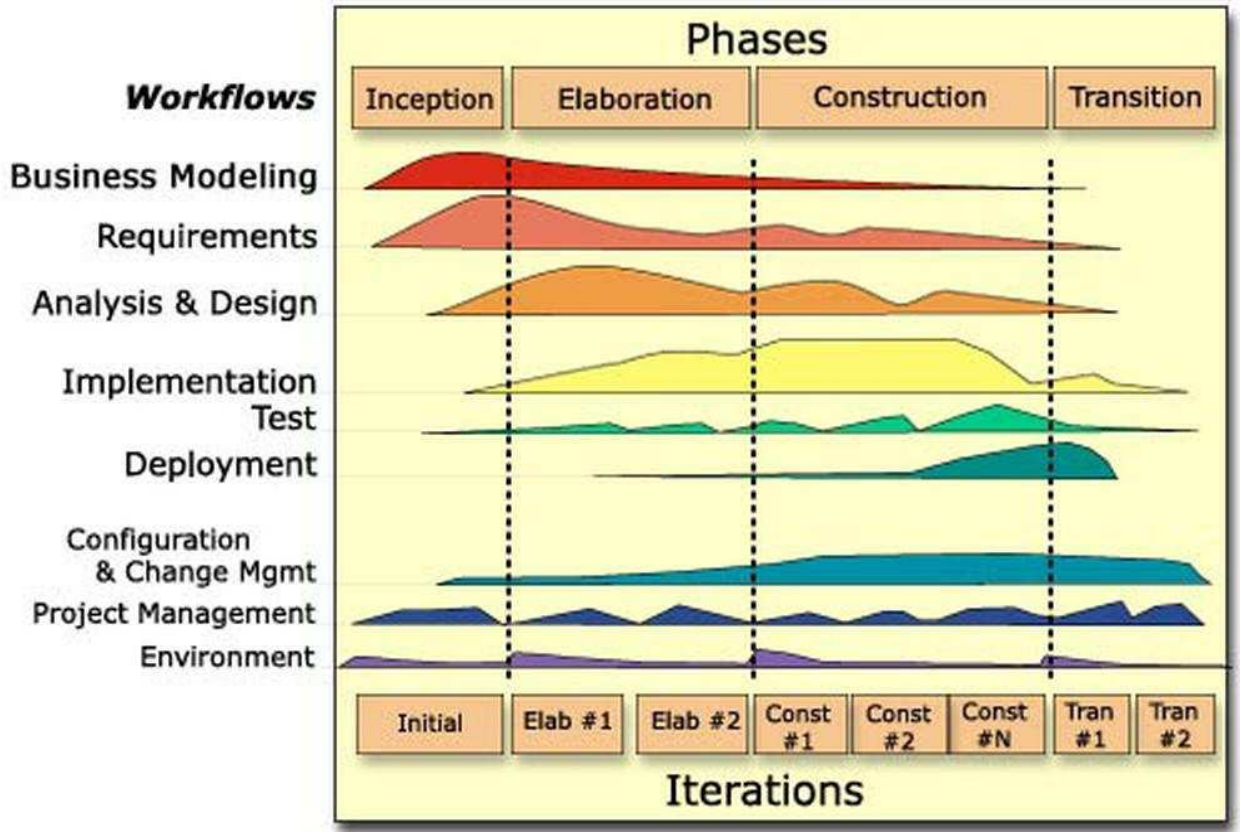
Watts S. Humphrey. Software Engineering Institute, CMU.

`http://portal.acm.org/citation.cfm?id=75122`

Software Processes (see notes)

- Waterfall (Royce)
- V Model (German Ministry of Defense)
- Prototyping
- Operational Specification
- Transformational (automated software synthesis)
- Phased Development: Increment and Iteration
- Spiral Model (Boehm)
- The Rational Unified Process (RUP)
- Extreme Programming (XP)

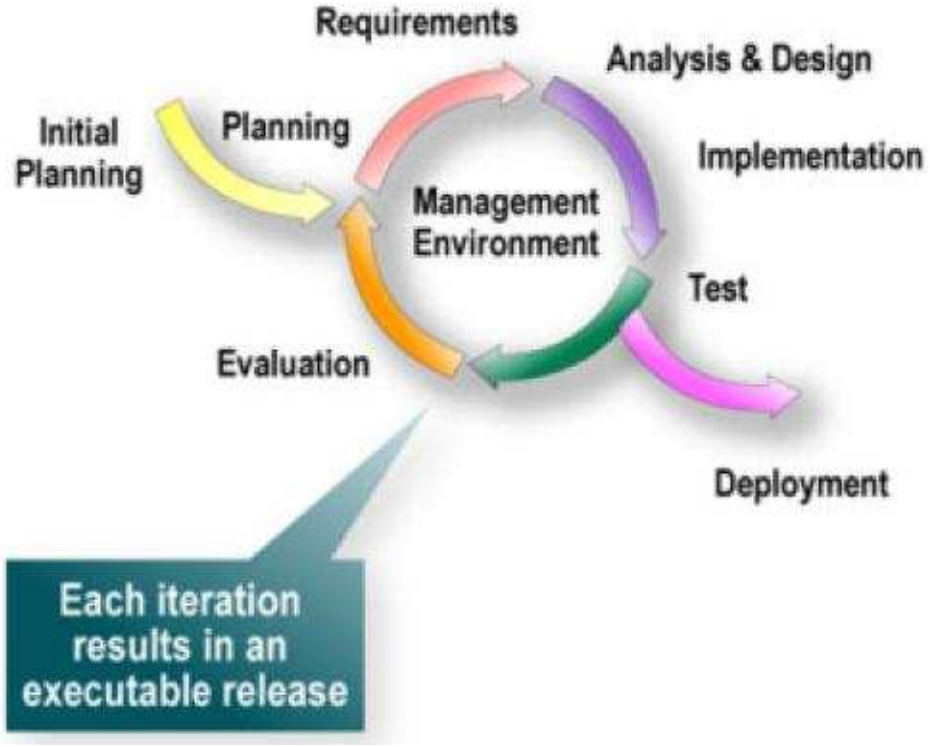
The Rational Unified Process (RUP): Activity Workload as Function of Time



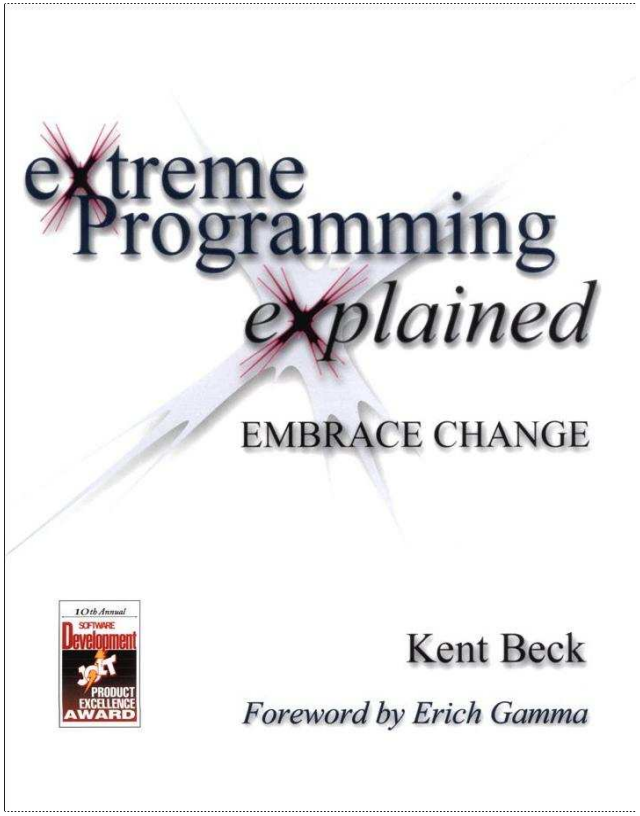
The Rational Unified Process (RUP): Observations

1. Waterfall-like **sequence** of
   Requirements, Design, Implementation, Testing.
2. Not pure waterfall:
   - **Phased Development** (**iterative**)
   - Overlap (**concurrency**) between activities
3. Testing:
   - **Regression** (test not only newly developed,
     but also previously developed code)
   - Testing starts **before** design and coding
     (Extreme Programming)

RUP: Phased Development



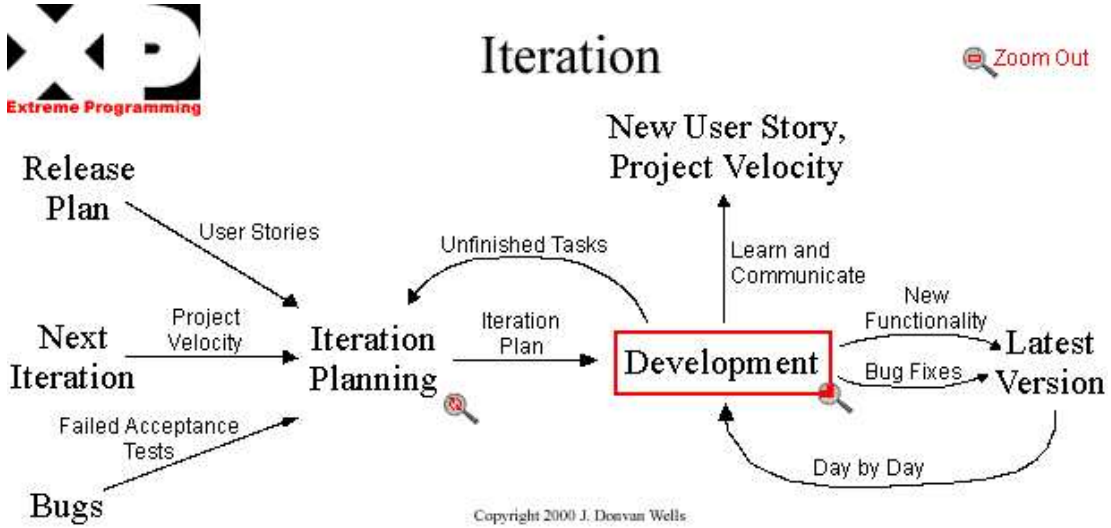
Extreme Programming (XP)



`www.extremeprogramming.org`

Extreme Programming (XP) highlights

- **User Stories** are written by the customers as things that the system needs to do for them. They drive the creation of acceptance **tests**.
- The project is divided into **Iterations**.



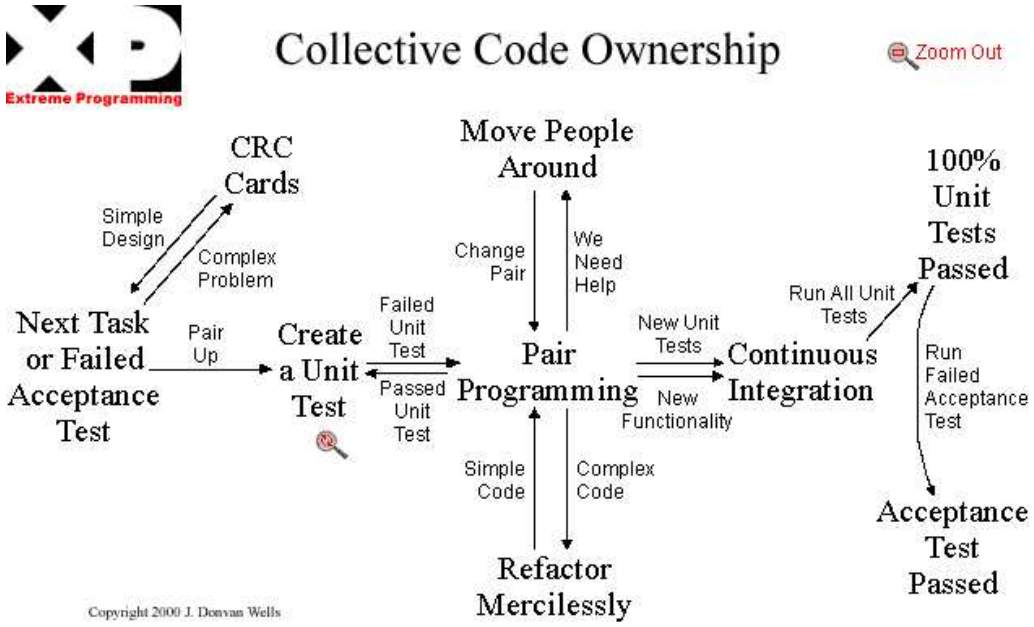
Extreme Programming (XP) highlights

Use Class, Responsibilities, and Collaboration **(CRC) Cards** to **design** the system.

Class Name:	
Superclasses:	
Subclasses:	
Responsibilities:	Collaborators

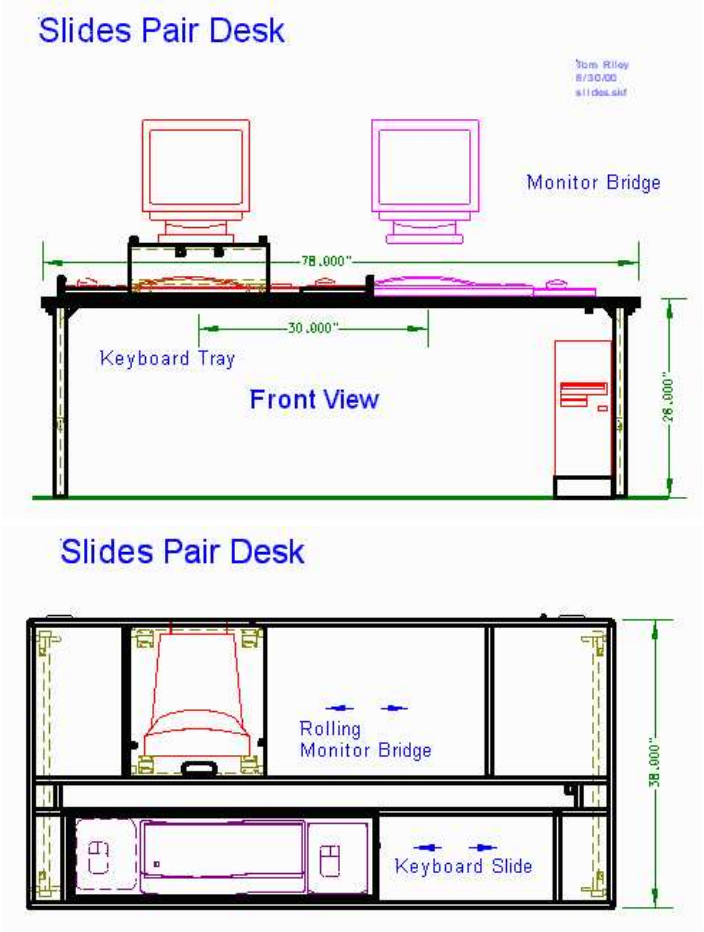
Extreme Programming (XP) highlights

- Code the Unit Test **first**.
- **All code** must have Unit Tests; All code must pass **all** unit tests before it can be released.



- **Refactor** whenever and wherever possible.

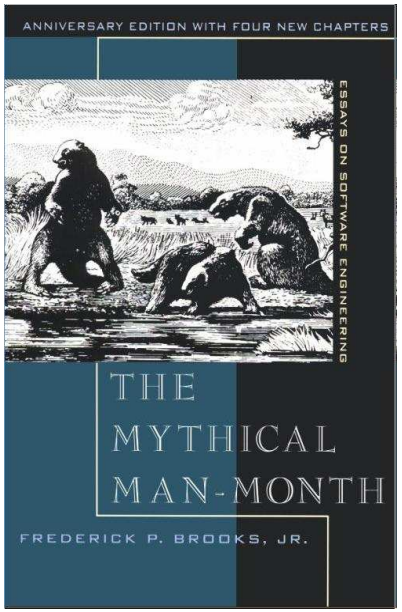
Extreme Programming (XP) highlights



Pair Programming

`www.charm.net/ jriley/pairall.html`

The Process influences Productivity

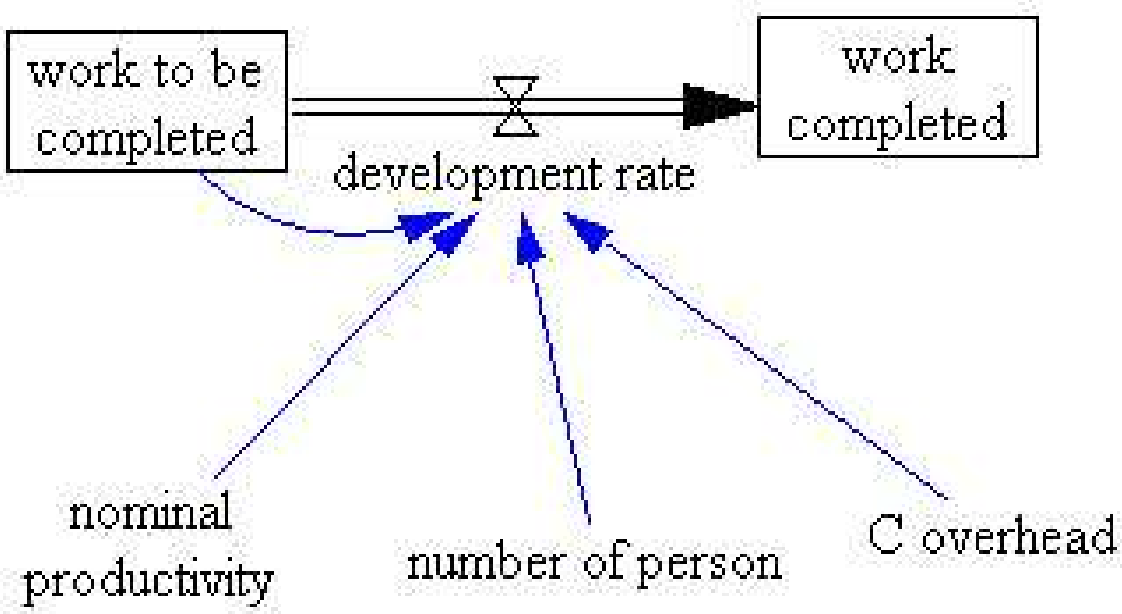


"Adding manpower to a late software project makes it later".

Fred Brooks. The Mythical Man-Month.

`http://www.ercb.com/feature/feature.0001.html`

Why Brooks' Law ? Team Size.

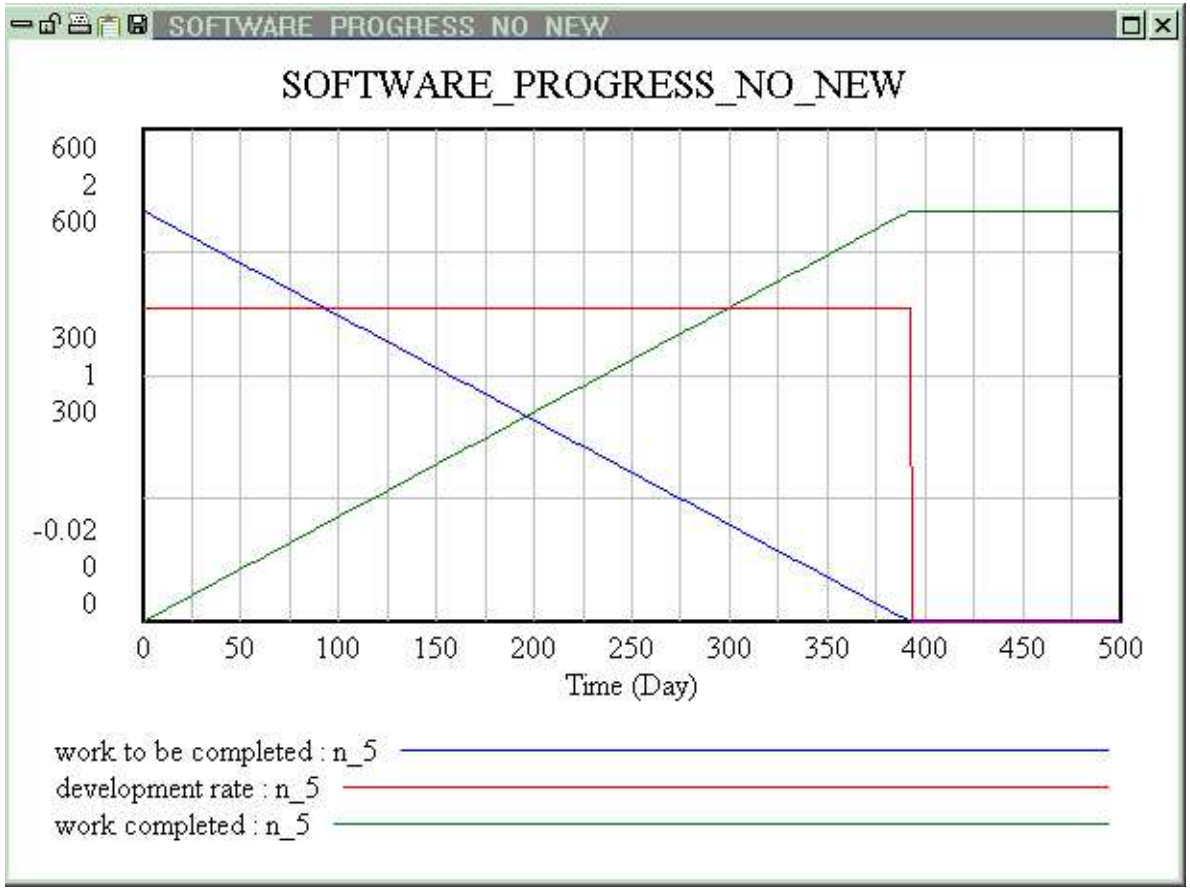


Model in **Forrester System Dynamics**

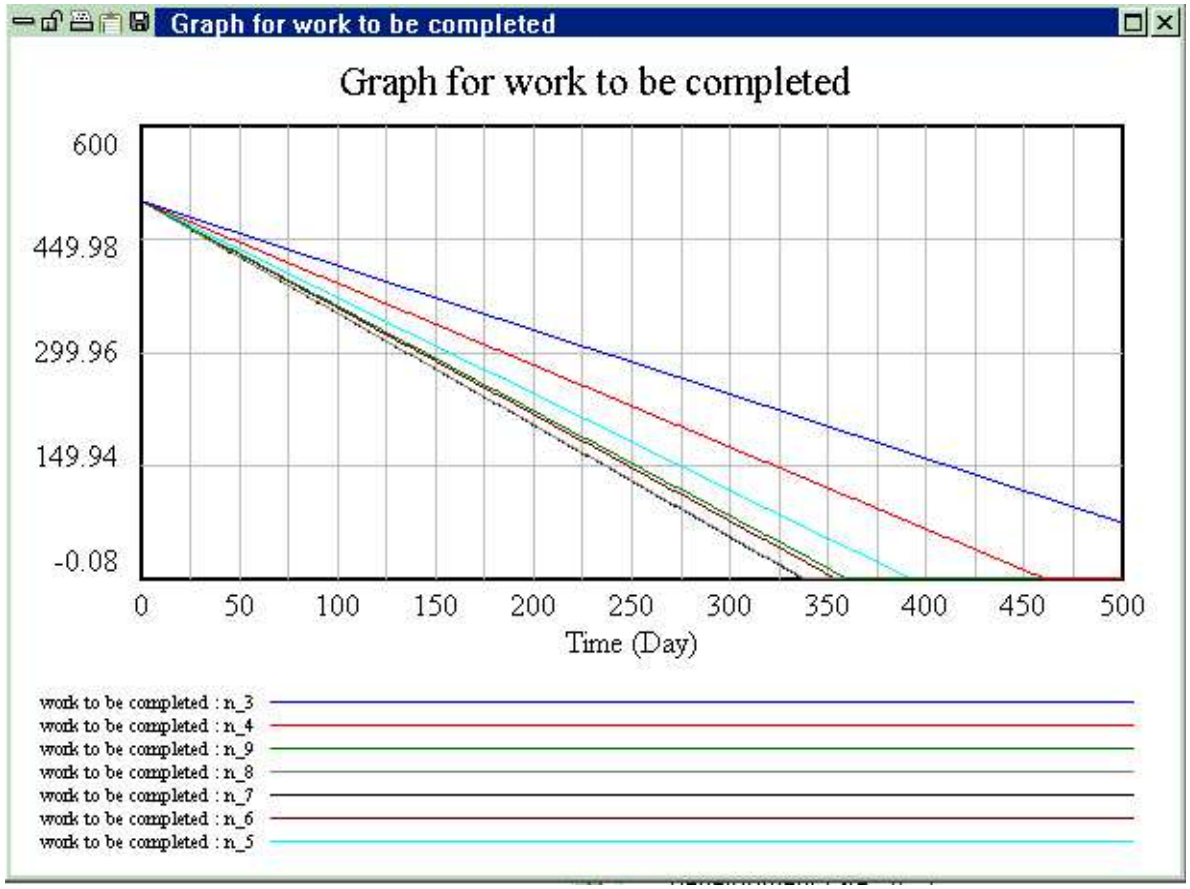
using `Vensim PLE` (www.vensim.com)

```
development rate =
 nominal_productivity* (1-C_overhead*(N*(N-1)))*N
```

Team Size N = 5

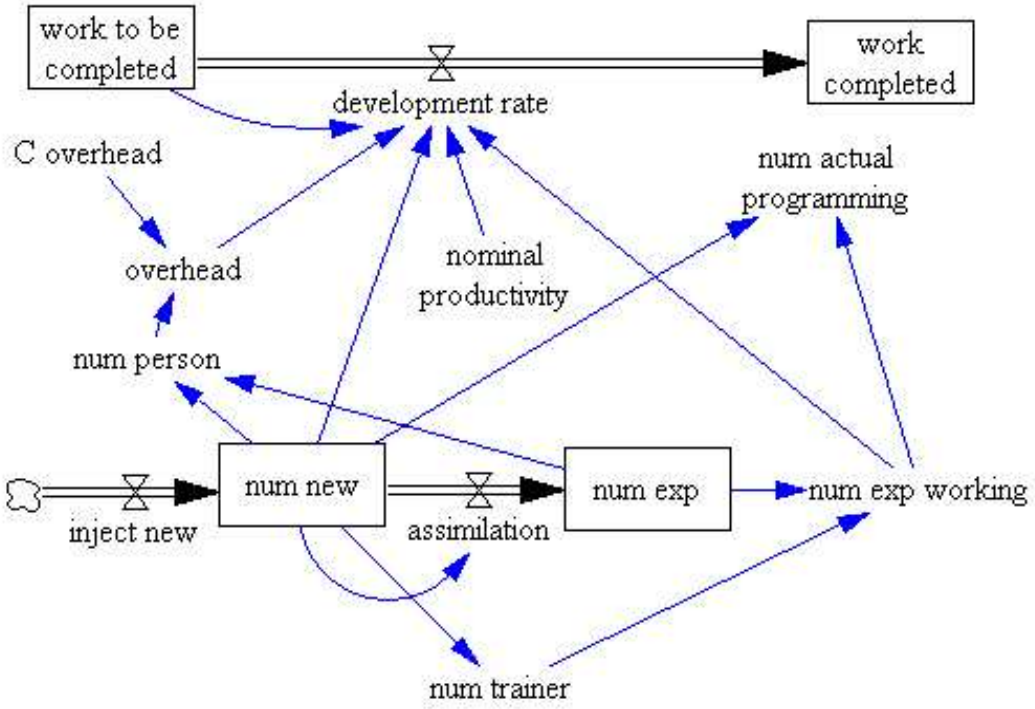

SOFTWARE_PROGRESS_NO_NEW
SOFTWARE_PROGRESS_NO_NEW
600
2
600
300
1
300
-0.02
0
0
0
50
100
150
200
250
300
350
400
450
500
Time (Day)
work to be completed : n_5
development rate : n_5
work completed : n_5

Team Size N = 3 . . . 9



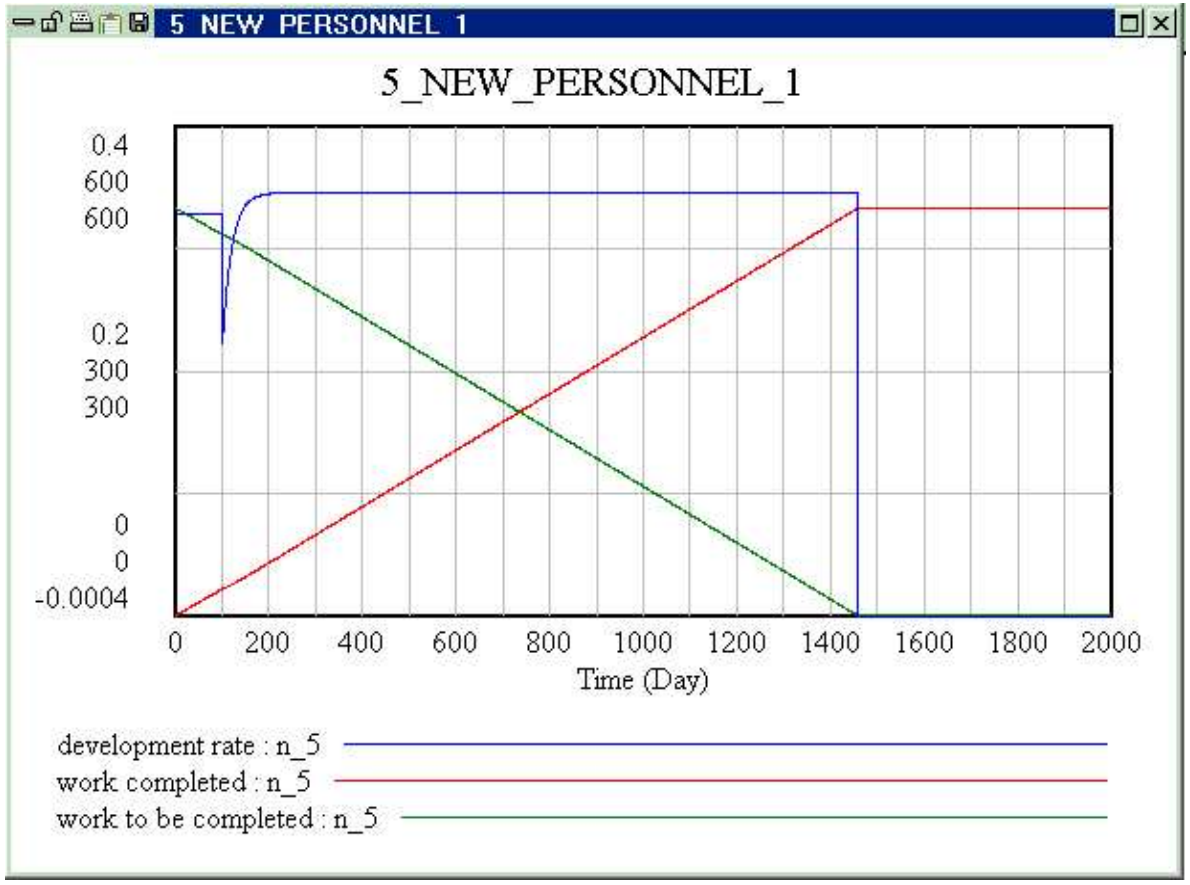
Optimal Team Size between 7 and 8

The Effect of Adding New Personnel (FSD model)

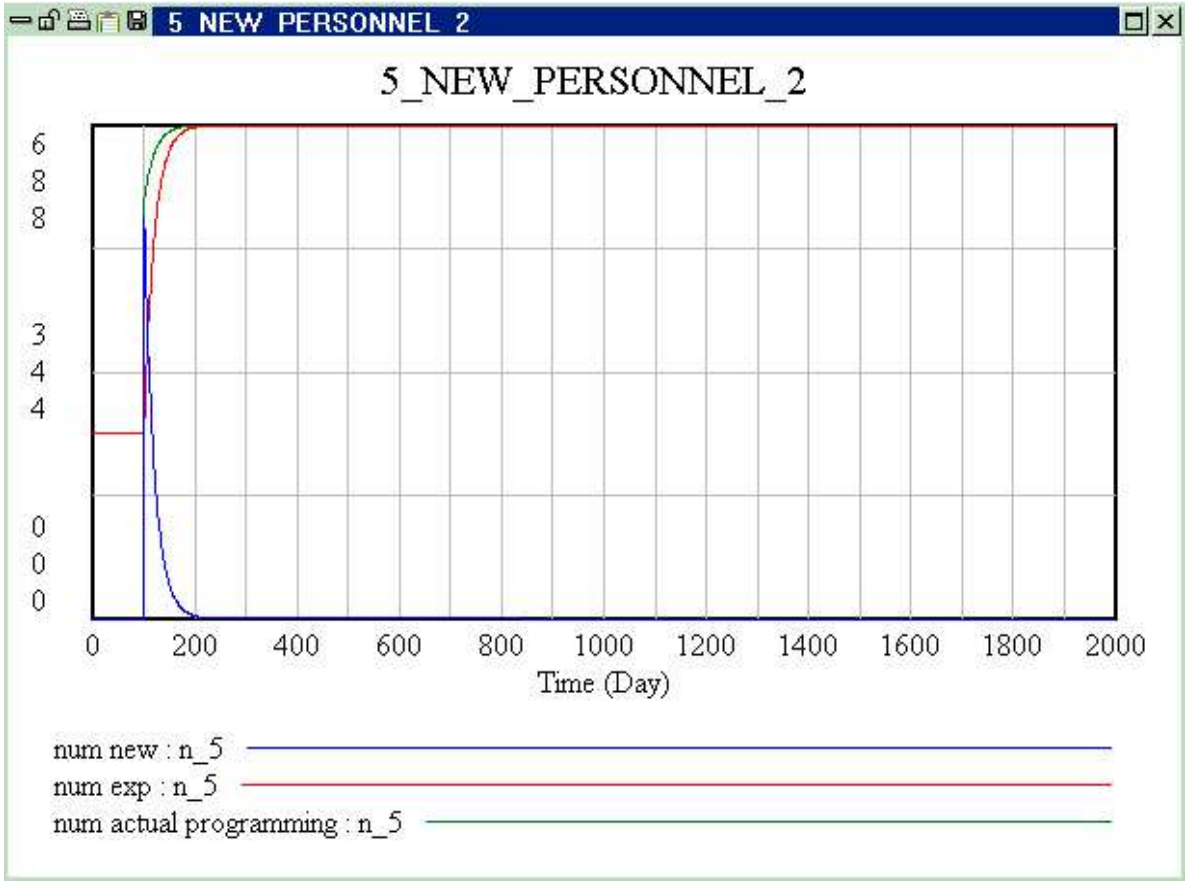



```
development rate = nominal_productivity*
 (1-C_overhead*(N*(N-1)))* (1.2*num_exp_working + 0.8*num_new)
```

5 New Programmers after 100 days



5 New Programmers after 100 days



0 . . . 6 New Programmers after 100 days