# System Dynamics Modeling for Overtime Management Strategy of Software Project

**Author: Jianguo Jia(**Tongji University)**, Xia Fan** (Fudan University)**, Yu Lu (**Tongji University)

401 Room, 43 Building, 99 Guoquan Road, Shanghai, China

Email: Jianguo.jia@alcatel-sbell.com.cn

**Abstract**

*Schedule overrun is a major problem that disturbed software project team. How to solve this problem? For most software project managers, the first reaction is to work overtime. There is no doubt overtime can alleviate this problem to some extend, but is it an effective way all the time? If not, when shall we give up overtime and change to other ways? This paper analyzed those problems in detail and gave some conclusions in the end. That is for a software project team which has reached its overtime limit, further overtime can only result in much longer completion date. Thus the best overtime policy is to first set a proper scheduled completion date, then try to find the minimums project completion time. The overtime range related to this minimums project completion time will be the critical point to stop further overtime.*

**Key Words**

Project Management, System Dynamics, Overtime

**Introduction**

Software industry can't get rid of its long exist "Software crisis" in spite of its high development. These crises are mainly shown as: schedule and cost overrun as well as poor product quality that do not meet the target customers' requirements. Among those, schedule overrun is one of the most frequently occurred problems. Almost every software project team has such experience more or less. It is generally believed that the reason why schedule tends to overrun lies in two aspects. One is the increasing complexity of software which makes it difficult for project manager to do initial evaluation at the early stage of the project when the amount of information is little. The other is ineffective decision-making of project managers when facing such problems. Since the first reason is external and hard to figure out, we can only focus on the second.

Currently, there are four policies for schedule slippages. That is overtime working, hiring new employees, extending completion date as well as reducing requirements, each should be taken under certain situation. There are already some papers in this special issue. For example, Abdel-Hamid, T.K.(1988) compared hiring new employees with extending completion date in his paper *The Dynamics of Software Project Staffing: A System Dynamics Based Simulation Approach* and drew a specific conclusion. That is in the early stages of the project when "Time Remaining" is generally much larger than the sum of the "Hiring Delay" and the "Average Assimilation Delay", the best policy is to hire new employees when schedule overrun occurred. With the project going on, When "Time Remaining" drops below 0.3 * (Hiring Delay + Average Assimilation Delay), the particular policy suggests that no more additions would be made to the project's workforce.

Schedule slippages at this late stage in the project would, thus, be handled through adjustments to the schedule completion date, and not through adjustments to the workforce level. When the "Time Remaining" on the project is between 0.3 and 1.5 times the sum (Hiring Delay + Average Assimilation Delay), both policies will be adopted. This paper gives us some useful implications but it ignores one important policy which is widely used in Chinese software development team to deal with schedule slippages. That is overtime working because it can be easily achieved and cost less. Little papers has ever touched on overtime issues till now. This may be due to the difference in national policy and culture. I will discuss about overtime issue at length in the following section by using Vensim software to set up dynamics models. To validate the model a real case of software development project named ISAM3.1 in Alcatel Shanghai Bell is also given.

**The structure analysis Introduction**

The model presenting in this paper was based on sample models in Vensim software. It involves four subsystems, namely: 1)The Human Resource Management Subsystem; 2)The Software Production Subsystem; 3)Overtime Subsystem; 4)Cost Subsystem. Fig.1 depicts the relations between the model's four subsystems.

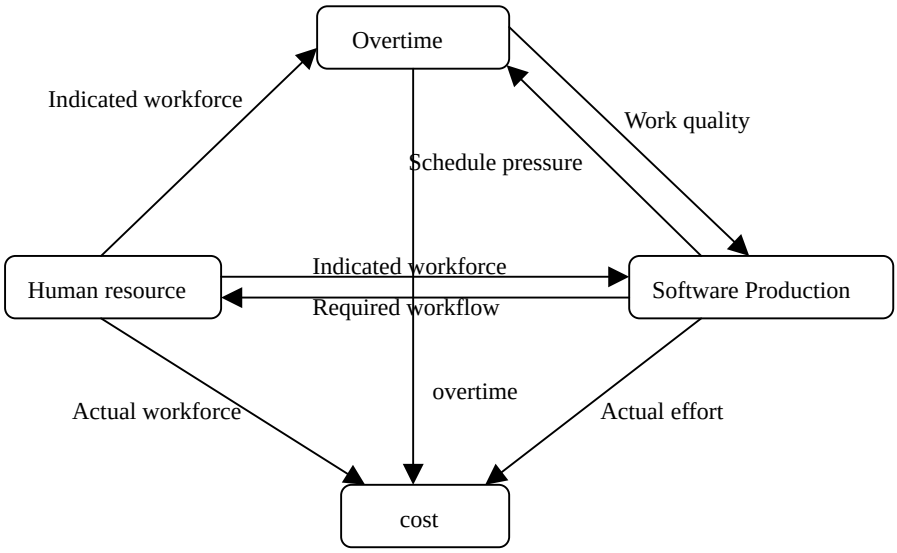



**Fig.1.** Relations between the model's four subsystems.

The software production subsystem involves develop, test and rework. It defined schedule pressure by "required workflow", "normal workflow" and "project is done". If project was not completed, "project is done" was set to zero, schedule pressure was equal to the result of normal workflow divided by required workflow. Here normal workflow is an constant, required workflow was defined by the result of scheduled time remaining divided by work remaining. Overtime subsystem made decision based on the parameter of schedule pressure delivered by software production subsystem and indicated workforce delivered by human resource subsystem, while Human resource subsystem adjust workforce level regarding required workflow. In the end, cost subsystem calculated total cost based on actual workforce multiplied by actual effort and overtime.

This section only gives a general introduction about the model structure; a more detailed description will be presented in the next section.

**Cause Loop Modeling**

When schedule pressure occurred, the first reaction for managers is to work overtime. It is generally accepted by overtime we can increase the amount of work completed thus decrease work remaining which will in turn reduce required workflow, eventually reduce schedule pressure. Obviously it is a negative loop also called balance loop. However it is often ignored that there exists another loop. That is overtime will not only increase the amount of work done, but inject more errors because of fatigue. In order to fix these errors, more work need to be done which will increase the work remaining and further increase required workflow, thus schedule pressure will also be increased. This positive loop will make schedule overrun even more serious. So it is very important in making proper overtime policy. The whole cause loop diagram was shown in fig.2.

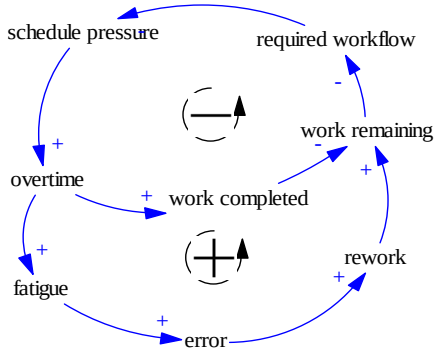



Fig.2. Causal loop

At the early stage, the negative loop has more influences than the positive one, so schedule pressure will be relieved. In other words, overtime is an effective way to solve schedule overrun problem. But with time going on, the influence of the positive loop will be strengthened. When its influences are greater than the negative one, the overtime is not longer useful in alleviate schedule pressure. It will even increase schedule pressure. At this moment further overtime must be prohibited and other decisions must be made.

**Dynamics Modeling and Assumption**

The model presented in this paper is based on two assumptions: One is that workforce level of the project is fixed. That is to say, workforce available for the project is limited. The assumption is made because our major concern is overtime issue. If workforce available is unlimited then there is no need to work overtime. When schedule pressure occurs, it can be relieved just by adding more workforces. Besides, in spite of the complexity of software production, we only focus on decision-making. So we abstracted software production process into develop, test and rework, without considering such details as requirements design, coding and so on.

Because the model is quite comprehensive and highly detailed, it is infeasible to fully explain it in the limited space of this paper. Since our major concern is overtime issue, we will, therefore, provide a detailed description for only the overtime subsystem. The overtime subsystem ,depicted in Fig.3 captured efficient work quality and efficient productivity which was effected greatly by average overtime. Here average overtime is a smooth of overtime.

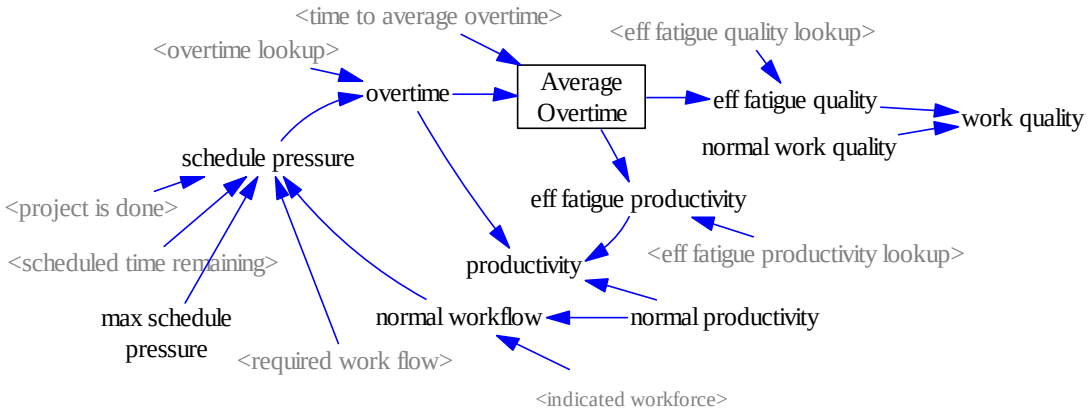



Fig.3. overtime subsystem

Major equations in this model are defined as following:
schedule pressure = IF THEN ELSE(scheduled time remaining<= 0 :AND: :NOT: project is done, max schedule pressure, ZIDZ(required work flow, normal workflow))
Productivity=overtime*eff fatigue productivity*normal productivity
Work quality=normal work quality*eff fatigue quality

**Base Run of the model**

Before we make the base run simulation, it is necessary to assign suitable initial values for those constants involved in the model. Table.1. lists the initial values for major constants based on a real software project called ISAM3.1 in Alcatel-sbell. All data is obtained by discussed with the related project manager and engineers.

Table 1. values of major constants

| **Constant name** | **Unit** | **Value** |
|---|---|---|
| Project size | KLOC | 76 |
| Scheduled completion date | Months | 10 |
| Max schedule pressure | Dimensionless | 5 |
| Max workforce | Persons | 20 |
| Normal productivity | KLOC/day/person | 1 |
| Normal work quality | Dimensionless | 0.9 |
| Quit time | Days | 0.5 |
| Time to transfer workforce | Days | 2 |
| Time to average overtime | Days | 2 |

Besides the lookup function of efficient fatigue productivity and efficient fatigue quality was depicted in Fig.4.

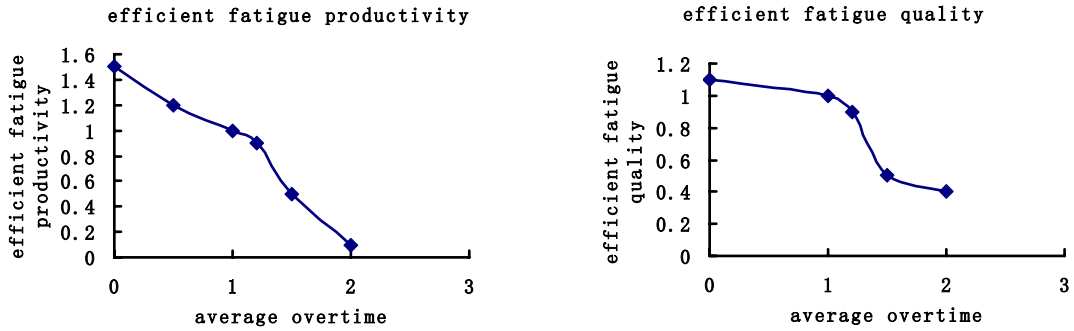
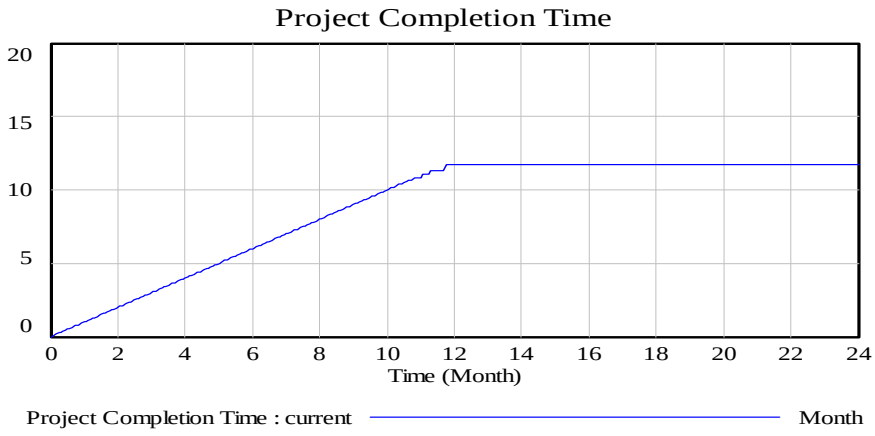
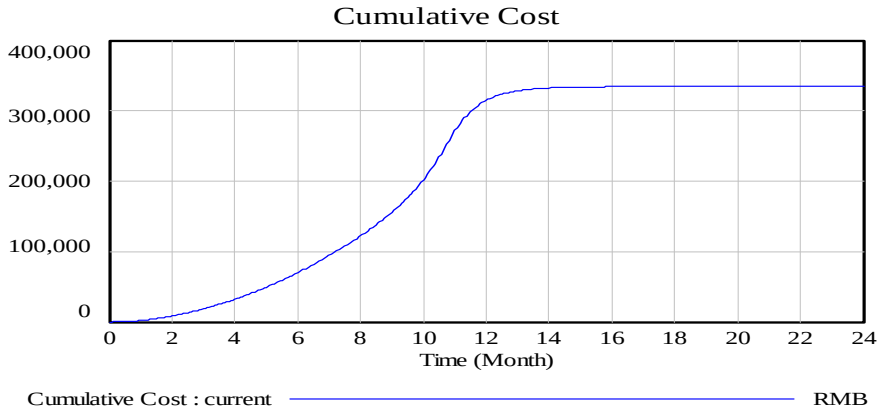



Fig.4.major lookup function description

Now we can make base run simulation with the specific parameters. Here we select four major variables: project completion time, cumulative cost, overtime, workforce to make a detailed analyze. The simulation result is shown below.

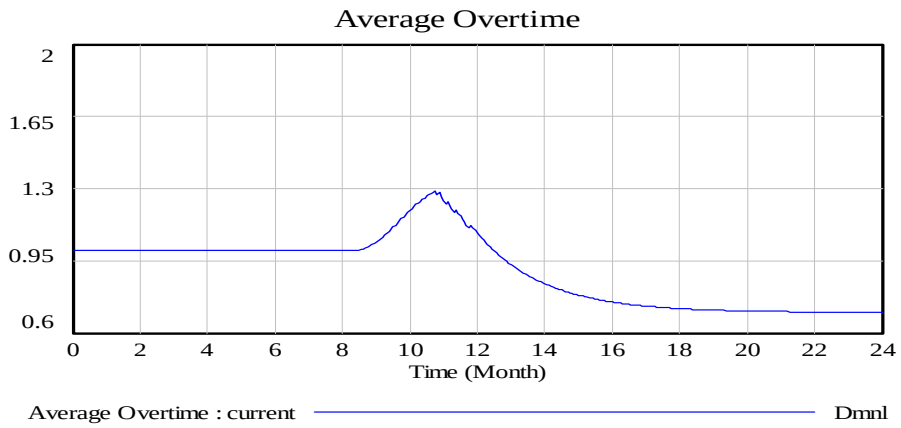



It can be seen from the base run result of project completion time, the project completion time first increased continuously until it reaches the maximum value. Here the maximum value is 11.75 months, which mean the project completed in 11.75 months, a bit later than the scheduled completion date.

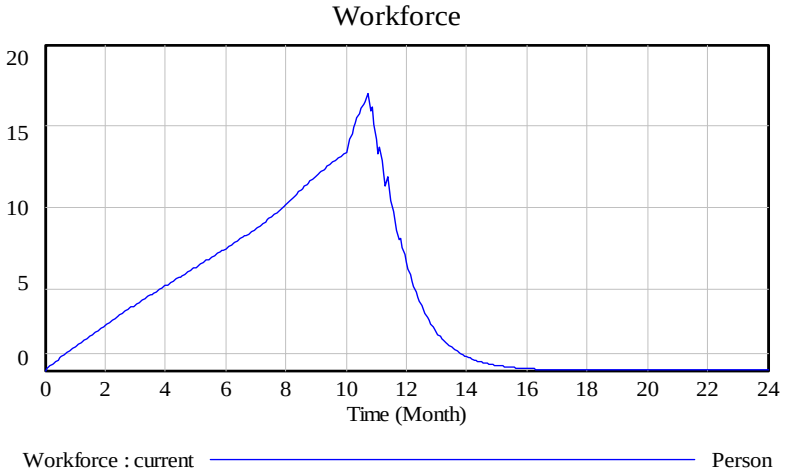



The tendency of cumulative cost is just like project completion time. First increased then maintained the level. The only difference is that it is not linear increased. This nonlinear increase is due to difference in overtime wages and normal wages. In reality overtime wages is usually two

times about normal wages.




The stick parts of the curve represent the overtime period. Here it starts in about eighth month and ends in about twelfth month which means the overtime period lasts for nearly four months.



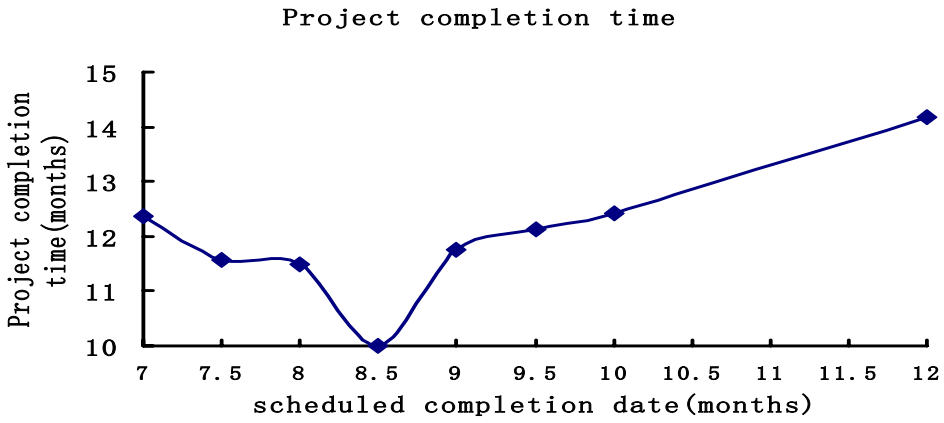

You may be surprised about the workforce curve. Why it is not a horizontal line since the previous assumption set it as a fixed value? In fact, in real software project team, engineers entered the team gradually, not at the same time. When the project is about to finish, engineers transfer activities is also gradually, not at once. This workforce curve reflects the dynamics changes of human resource in a real software project team. So it is reasonable.

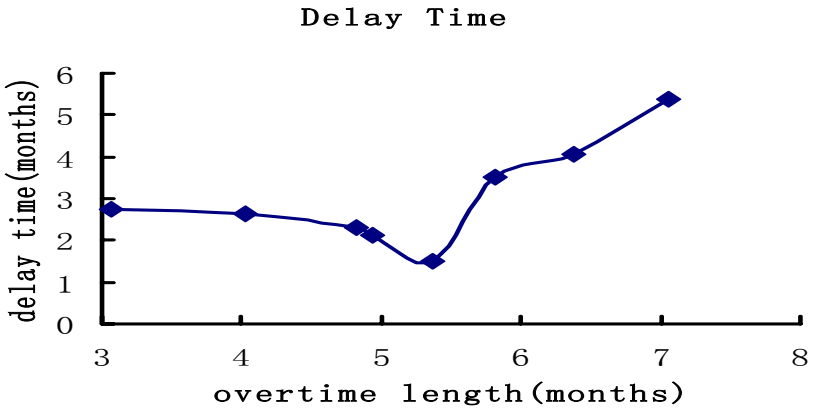
**Scenario planning and modeling**

To make further analysis, we set scheduled completion time to different values and run related simulations. After each simulation, write down the values of selected variables. The final result is shown in Table 2.

| Run | Scheduled completion date (months) | Project completion time (Months) | Delay Time (Months) | Cumulative cost (RMB) | Overtime length (Months) |
|---|---|---|---|---|---|
| 1 | 7 | 12.37 | 5.37 | 428,635 | 7.06 |
| 2 | 7.5 | 11.56 | 4.06 | 397,463 | 6.38 |
| 3 | 8 | 11.5 | 3.5 | 373,498 | 5.81 |
| **4** | **8.5** | **10** | **1.5** | **355,384** | **5.37** |
| 5 | 9 | 11.75 | 2.75 | 336,672 | 4.94 |
| 6 | 9.5 | 12.14 | 2.64 | 312,456 | 4.82 |
| 7 | 10 | 12.43 | 2.43 | 287.432 | 4.02 |
| 8 | 12 | 14.18 | 2.18 | 231.421 | 3.06 |

We can see from the table the change tendency of project completion time, cumulative cost and overtime length corresponding to the increasing of scheduled completion date. Here we use the constant scheduled completion date to represent schedule pressure. With the increasing of scheduled completion date, schedule pressure will be reduced. It can be observed from the table project completion time first go down and then go up. There exists a extremism point. This phenomenon was due to the interaction of negative loop and positive loop. Cumulative cost and overtime length remain decreased with less schedule pressure. Another important implications in this table is that when overtime length achieved a certain point (in the table is 5.37), further overtime can only result in more delay. As can be seen from run 3 to run 1, overtime length increased, but delay time is also increased. Fig.5 will be given to illustrate the above conclusion more clearly.






From this two charts, we can make a best overtime policy under the specific situation, that is to set schedule completion date to 8.5 months, when overtime length arrives 5.37 months. Further overtime should be forbidden, because at this moment more overtime will only make the schedule more behind.

**Conclusion**

Till now, we have made a rather detailed analysis on overtime policy in software development project and drew two useful conclusions. One is that overtime is not always an effective way to reduce schedule pressure, when the team has reached its overtime limits, further overtime can only result in more behind schedule. The other is to make a best overtime policy, we first need to set a proper scheduled completion date based on initial project definition, and then try to find the minimum project completion time through system dynamics modeling. The overtime range related to this minimum project completion time is the critical point to stop further overtime.

**Reference**

[1] T. K. Abdel-Hamid, “The dynamics of software development project management: An integrative system dynamics perspective,” Ph.D. dissertation, Sloan School of Management, MIT. Jan. 1984.

[2] Abdel-Hamid, T.K, The Dynamics of Software Project Staffing: A System Dynamics Based Simulation Approach. IEEE transactions on software engineering, 1989, 15(2), 109-119.

[3] Abdel-Hamid,T.K, On the Utility of Historical Project Statistics for Cost and Schedule Estimation. Journal of Systems and Software, 13:71-82, (1990).

[4] Wang qifan, System Dynamics [M]. Beijing, Tsinghua University Press，1994.

[5] Alexandre Rodrigues. The role of system dynamics in project management [J]. International Journal of Project Management, 1996, 14(4): 213-220.

[6] Putnam,L.H., Myers,W., 1996.Executive Briefing Controlling Software Development. IEEE Computer Society Press, Silver Spring, MD.

[7] Rodrigues,A.G., Williams,T.M.,1997.System dynamics in software project management: towards the development of a formal integrated approach.Eur.J.Inf.Syst.6,51-56.

[8] David Ford, John Sterman. Dynamics modeling of product development processes [J]. System Dynamics Review, 1998, 14(1); 31-68.