A Software Process Modeling Approach for Providing Consistency between the Process Description and Simulation

박승훈* · 최경식** · 배두환***

1. Introduction

Software Process Modeling (SPM) researches can be classified into software process description, software process enactment, and software process simulation. Most of the SPM researches have focused on describing a software process for understanding and communication and supporting for execution or enactment of the process within Process-centered Software Engineering Environment (PSEE) using Process Modeling Langauges (PMLs)[1-4]. On the other hand, software process simulation modeling researches have concentrated on a variety of management issues such as strategic management, planning, and control[5]. In general, the process simulation model is created independently with the process description model. In this context, although most of PMLs support the consistency between process description and enactment, there is little consideration for the consistency between process description and simulation. Inconsistency between the process description and simulation forces the process engineers to organization has a detailed process description model.

In this paper, we propose an approach to extracting the process simulation model from the process description model, providing the consistency between the process models. A process description model is created using Software Process Engineering Metamodel (SPEM), which is a metamodel for defining processes and their components as a standard for process modeling[6]. A metamodel enables the process description model to be transformed into the process simulation model consistently by specifying the elements of the two process models and relationships between the elements which should observe during transformation.

SPEM is extended to support for process simulation. We express it as *the extended SPEM*. Once a process model is described based on the extended SPEM, the process sim-

※ 교신저자(Corresponding Author) : 박승훈, 주소 : 대전시 유성구 구성동 373-1 한국과학기술원 전산학과(305-701), 전화 : 042)869-5579, FAX: 042)869-8488, E-mail : seunghun@se.kaist.ac.kr
* 한국과학기술원 박사과정
** 육군 소령
(E-mail : kschoi48@gmail.com)
*** 한국과학기술원 전산학과 교수
(E-mail : bac@se.kaist.ac.kr)
※ 본 논문은 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성, 지원사업의 연구결과로 수행되었음

ulation model can be automatically created from the process description model. We focus on deriving a Software Process Simulation Model (SPSM) from the process model described by the extended SPEM, providing the consistency between the process description and the process simulation. Discrete Event System Specification (DEVS)-based software process simulation modeling [7] is applied in the simulation.

This approach allows process engineers to reduce the effort for developing a software process simulation model by extracting the model from a software process description model. The process models can keep the consistency between the models using a transformation mechanism without modification. This approach also makes the process change time, redesigning the organization's process model based on the feedback from simulation results, to be reduced. Process engineers can develop the process simulation model in a more efficient and convenient way by deriving the model from the process description model based on the extended SPEM, concentrating on high-level process modeling.

The structure of this paper is as follows: In Section 2, we briefly introduce SPEM and the DEVS formalism. Section 3 and Section 4 describe how to extend SPEM and how to extract DEVS-based SPSM from the extended SPEM-based model, respectively. Section 5 provides a case study for this approach.We make a software process description model of ISPW-6 example process with the extended SPEM and convert it to DEVS-based SPSM. Finally, Section 6 summarizes the main results of this paper and gives a plan for future work.

2. Background

2.1 SPEM overivew

The SPEM, defined by Object Management Group (OMG), is used to describe a concrete software development process or a family of related software development processes[6]. SPEM uses an object-oriented approach to modeling a family of related software processes and SPEM uses UML as a notation. It's a four-layered architecture of modeling as defined by OMG. The SPEM specification is structured as a UML profile, i.e., a set of stereotypes, tags, and constraints added to the UML standard semantics. The SPEM is built from the SPEM Foundation package, a subset of UML 1.4, and the SPEM Extensions package, which adds the constructs and semantics required for software process modeling. At the core of SPEM is the idea that a software development process is a collaboration between abstract active entities, called ProcessRoles, that perform operations, called Activities, on concrete, tangible entities, called WorkProducts[8].

2.2 DEVS formalism

DEVS is a general formalism for discrete event system modeling based on the set theory [9]. It allows representing any system by three sets and four functions: Input Set, Output Set, State Set, External Transition Function,

Internal Transition Function, Output Function, and Time Advanced function. DEVS formalism provides the framework for information modeling which gives several advantages to analyze and design complex systems: Completeness, Verifiability, Extensibility, and Maintainability [10].

DEVS has two kinds of models to represent systems. One is an atomic model and the other is a coupled model which can specify complex systems in a hierarchical way[9]. The DEVS model processes an input event based on its state and condition, and it generates an output event and changes its state. Finally, it sets the time during which the model can stay in that state. An atomic DEVS model is defined by the following structure[9]:

$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta \rangle$$

where:

- X is the set of input values,
- Y is the set of output values,
- S is the set of states,
- $\delta_{ext} : Q \times X \rightarrow S$ is the external transition function,
  where $Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$ is the total state set, e is the time elapsed since last transition
- $\delta_{int} : S \rightarrow S$ is the internal transition function,
- $\lambda : S \rightarrow Y$ is the output function,
- $ta : S \rightarrow R+$ is the set positive reals bet.

The behaviors represented by four functions of the atomic model are as follows:

- An atomic model can stay only in one state at any time
- The maximum time to stay in one state without external event is determined by $ta(s)$ function
- When an atomic model is in a state $(0 \leq e \leq ta(s))$, it changes its state by δ_{ext} function if it gets an external event
- If possible remaining time in one state is passed $(e = ta(s))$, it generates output by function and changes the state by δ_{int} function

DEVS coupled model is constructed by coupling DEVS models. Through the coupling, the output events of one model are converted into input events of other models. In DEVS theory, the coupling of DEVS models defines new DEVS models (i.e., DEVS is closed under coupling) and then complex systems can be represented by DEVS in a hierarchical way.

3. Extending SPEM to support process simulation

3.1 Overview of our approach

The overview of our approach is shown in Figure 1.

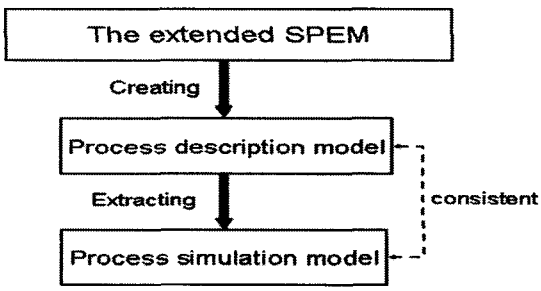

Figure 1. Overall approach

A metamodel describes the modeling elements and the rules or constraints which are allocated to the relationships between the modeling elements. By allowing SPEM to contain process modeling elements for process simulation and relationships between the elements for process description and simulation, we can provide the consistency between process description and simulation. A process description model is described based on the extended SPEM. The process description model contains the process modeling elements such as activities, workproducts, and roles for describing the process. A process simulation model is extracted by transforming the process modeling elements into the simulation modeling elements of DEVS and adding quantitative relationships among the elements. The process modeling elements are transformed into the simulation modeling elements by mapping between them. The mapping is performed for three types of elements: structural modeling elements, behavioral modeling elements, and quantitative modeling elements. The DEVS-based SPSM is generated by the mapping and then we integrate quantitative simulation equations and parameters into the DEVS-based SPSM.

3.2 The extension of SPEM

In this subsection, we describe how to extend SPEM for extracting the process simulation model from the process description model. We extend SPEM to provide the consistency between the process models. Figure 2 shows the part of the extended SPEM. We extend the main structural elements for process description which are defined in ProcessStructure package of the SPEM Extensions package[6]. However, we restrict ourselves to stay as close as possible to standard SPEM to minimize the technical complexity.

The elements of the process simulation model are a process model, a simulation engine, and quantitative data. The process model provides the overall structure and execution sequence of the simulation. The simulation engine executes the process defined by the process model, advances the simulation time (e.g., activity duration), and calculates the simulation parameters which users want to know (e.g., effort). The process model should contain the parameters for simulation to execute the simulation of the process model. We, therefore, allow process engineers to specify the parameters into activities. Activity is extended by adding the parameters such as duration, effort, and state as its attributes. A simulation engine consumes not roles but resources assigned to the roles. By adding Participant to the metamodel, the simulation engine can recognize a person, system, or

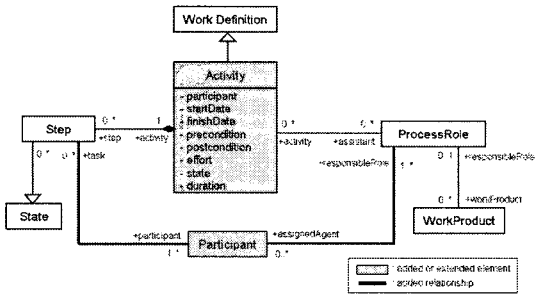

Figure 2. Part of the extended SPEM

team that is assigned to the activities during simulation.

4. Deriving DEVS-based SPSM from the extended SPEM-based process description model

4.1 Mapping the extended SPEM to DEVS

For deriving DEVS-based SPSM from the process description model we first map the process description modeling elements of the extended SPEM to DEVS to enable the DEVSim simulation engine, which is a C++ based DEVS simulation environment, to execute the software process. We devise the mechanism to incorporate the representative process performance variables, such as duration, effort into the derived DEVS-based SPSM. The duration of the process is calculated by the time advance function of the DEVS and the effort is calculated by the resource model which is defined by the role assignment to activities. Whenever the activity of the process advances the time by the time advance function, the simulation model calculates the effort consumed. Undoubtedly, we need a more detailed process simulation model to precisely analyze the process and other process performances, but this approach provides process modelers with basic structure and functionality of simulation program with less effort. In the remaining part, we present the process of the model transformation. We map the process description modeling elements to modeling elements of DEVS as shown in Figure 3.

	The extended SPEM	DEVS
Structural Modeling Elements	<<Process>> <<WorkDefinition>>	Coupled model
	<<Activity>>	Atomic model
	<<WorkProduct>> <<Document>>	Input, output event
Behavioral Modeling Elements	<<Step>>	State set, State transition function, Output function
Quantitative Modeling Elements		Time advance function, State transition function

Figure 3. Mapping table of the extended SPEM to DEVS

We extract structural modeling elements (i.e., input, output, coupling, etc.), behavioral modeling elements (i.e., state transition functions, output function), and quantitative modeling elements (i.e., time advance function) from the extended SPEM for process simulation.

The structural modeling elements include Process, WorkDefinition, Activity, WorkProduct, Document, etc. The Activity, the most basic element for process description, describes a piece of work performed by the ProcessRoles: the tasks, operations, and actions that are performed by a role. We map Activity to the atomic model which is a basic modeling element in DEVS. Process is a complete description of a software process, which includes WorkDefinitions, Activities, ProcessRoles, and WorkProducts. The WorkDefinition is the work performed in the process and can describe the Work-Breakdown Structure (WBS)[6]. One WorkDefinition is composed of another and has Activities as a subclass. Process and WorkDefinition are mapped to the coupled model,

which couples atomic models together to model a complex system hierarchically. The Work-Product is anything produced, consumed, or modified by an Activity in a process. It may be a piece of information, a document, a model, a source code, and so on. This is mapped to a set of input and output events which trigger the execution of atomic models.

The behavioral modeling elements include Steps. The Step, an atomic element of an Activity, describes the behavior of the Activity. We map the Step to state sets, state transition functions, and output functions. One Activity is decomposed into several Steps, which are mapped to state sets, and interacts with other Activities by sending and receiving Work-Product. The interactions occurred in one Activity are mapped to state transition functions and output functions, which specify how to change its state and make outputs based on the inputs and internal constraints.

The quantitative modeling elements are incorporated in the Steps. The TimeEvent of the State Machines in UML triggers the state transition[11], which is mapped to the time advance function. Based on the returned time from the time advance function, the model calculates the duration of each activity, triggers the state transitions, and generates outputs. The equations for calculating process performance parameters derived from the historical project data are mapped to the internal state transition function.

4.2 Integration of quantitative information into the DEVS-based SPSM

The mapping enables the extended SPEM-based software process model to be run by DEVSim simulation engine. However, we need more detailed mapping mechanism. The important point is how to integrate quantitative simulation equations and parameters into the DEVS-based SPSM. The most typical performance variables of the process simulation are duration (schedule) and effort (cost). We define the relationships between modeling elements of the extended SPEM and DEVS to incorporate those process performance variables.

The performance variables are computed via state transition, which is represented in the format below[11]:

event-name (parameter-list)
[guard-condition]/action-expression

The duration of the Activity is modeled as a TimeEvent using the following format[11]:

after(TimeForActivity)

The keyword after represents that when the time since the entry to the current state is elapsed to the amount of "TimeForActivity", the TimeEvent is occurred and triggers any transition that depends on the event. We map this into the time advance function of DEVS. The time advance function returns the duration time of each activity. The effort (e.g., Person-Months) is modeled as a resource model, which

is composed of FIFO (First-In First-Out) Queue models. The FIFO Queue model, an atomic model, represents ProcessRole which is assigned to Activities. The multiplicity attribute of the Perform association between ProcesRole and Activity represents the size of the FIFO Queue and the connection of the association defines the coupling relationships between FIFO Queue model and the Activity. Figure 4 illustrates relationships between Activity and FIFO Queue. When the Activity is being executed, it requests the assigned ProcessRole in the Resource model and then the FIFO Queue generates an output and reduces its size. If the FIFO Queue is empty, the Activity can't proceed and waits until the FIFO Queue is available. On the other hand, when the Activities are scheduled to perform at the same time, we can assign a priority on each of them, which is modeled with the Precedes dependency defined in SPEM.

Other performance variables not defined here are modeled as an attribute and operation of WorkProduct or Document. The attribute and operation are mapped to attribute and operation of DEVS Message, which is used for transferring the information of one model to another.

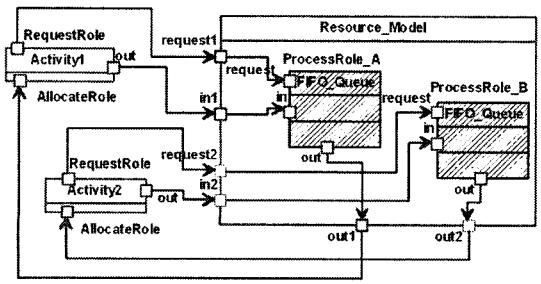

Figure 4. Resource model for effort calculation

This enables the software engineers to have extensibility and flexibility to implement the quantitative simulation model.

4.3 Implementation of the translator

The implementation of the translator is relatively straightforward. Most of UML tools support XMI for interchanging models in a serialized form. We analyzed the exported XML files and defined transformation rules of Extensible Stylesheet Language Transformations (XSLT). We extract all the necessary information discussed in this section through XSLT processing and finally generate DEVSim classes by Document Object Model (DOM) processing. We have done the analysis on the exported XMI and defined the XML schema, and have transformed it using Xerces[12] parser.

5. Case Study

We have performed a case study to evaluate whether the transformation provides the consistency between the process description model and the process simulation model. We have modeled ISPW-6 example process[13] with the extended SPEM and transformed it to DEVS-based SPSM. We use commercial UML modeling tool, Enterprise Architect[14], which supports UML Profile for SPEM, to model ISPW-6.

We will present the process description model based on the extended SPEM of the ISPW-6 with several UML diagrams which contain

SPEM icons [6] to provide notational convenience. The model is transformed to the DEVS-based SPSM, which uses the point estimate data used in [15] for the values of the simulation parameters. This scenario is preplanning of the ISPW-6 process under no resource constraints. Finally, we will demonstrate the transformed simulation program of the DEVS-based SPSM.

5.1 SPEM-based software process description model of ISPW-6

The UML profile for SPEM gives benefits of using UML diagrams to present different perspectives of a software process model: in particular, Class diagram, Package diagram, Activity diagram, Use Case diagram, Sequence diagram, and Statechart diagram[6]. We use three UML diagrams among them: Use Case diagram, Activity diagram, and Statechart diagram. We create two Use Case diagrams. Instead of showing them, we describe the diagrams. A Use Case diagram describes the decomposition of the Develop Change Test Unit Phase into several WorkDefinitions and Activities. At first, Develop Change Test Unit Phase is decomposed into Project Magmt, DevCode, Dev Test, Test Unit WorkDefinitions. Each of WorkDefinitions is decomposed into serveral Activities. For example, the Dev Test WorkDefinition is decomposed into Modify Test Plans Activity and Modify Unit Test Package Activity. WorkDefinitions and Activities would be transformed to the coupled model and the atomic model of DEVS, respectively. For example, the Dev Test WorkDefinition is transformed to the coupled DEVS model which include the atomic models of the Modify Test Plans Activity and Modify Unit Test Package Activity. The other Use Case diagram describes ProcessRoles involved in the process and their assigned Activities. This is transformed to resource model, which calculates the effort for the process.

Figure 5 shows the inputs, outputs, and sequences of Activities and the swimlane represents ProcessRoles assigned to each Activity for ISPW-6. This diagram gives the structural information of the process. For example, an atomic model, Modify Design Activity, outputs Modified Design message, which is connected to the Review Design atomic model. This shows the internal coupling among atomic models and the message transfer.

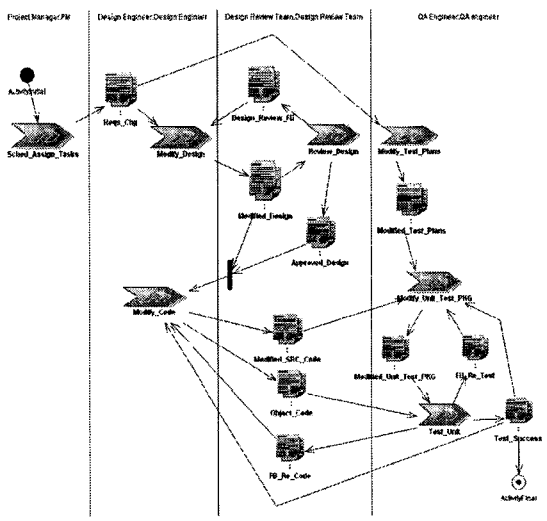

Figure 5. Activity diagram for sequence of Activities, Inputs, and Outputs

5.2 DEVS-based SPSM of ISPW-6

Describes overall structure of the DEVS-based SPSM. Compared to Figure 5, the whole Activities and WorkProducts are transformed into each of atomic models and input/output events of Figure 6. The ExperimentalFrame model, which is a predefined DEVS model, plays a role of a measurement system or observer like an oscilloscope in electronics. It generates inputs, Reqs Chg in this model, to trigger the ISPW-6 process model and accepts and analyzes the simulation results.

The snapshot of the implemented simulation program is shown in Figure 7. It is composed of scenario input, simulation results display, simulation analysis, and simulation control part. The simulation results and analysis for project management are out of the scope for this paper, and you can reference[5]. When process engineers want to change the organization's process with the simulation results, they change the process description model, transform the model,

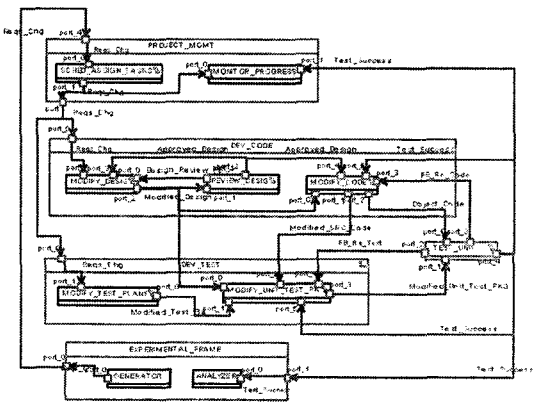

Figure 6. Overall architecture of the DEVS-based SPSM of the ISPW-6 process

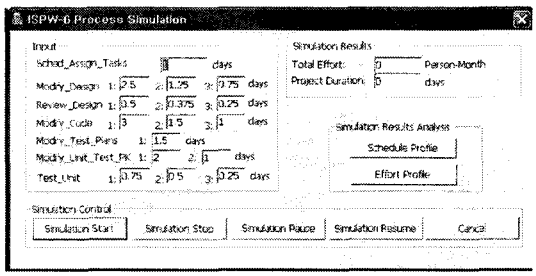

Figure 7. DEVSim program of ISPW-6

and simulate the transformed model iteratively. Process engineers can easily evaluate the alternatives of the organization's process using simulation.

6. Conclusion and Future Work

We proposed an approach to extracting the process simulation model from the process description model, preserving the consistency between the process models.

We extended SPEM to provide the process models with the consistency. We also provided the transformation mechanism to extract a process simulation model from a process description model and evaluated the mechanism by demonstrating a case study.

This approach allows process engineers to reduce the effort for developing software process simulation. Process engineers also develop the process simulation model more efficiently and conveniently because the process simulation model is derived from the process description model based on the extended SPEM. This approach can contribute to a software process improvement by reducing the feedback time of redesigning the organization's process

model based on a simulation.

As a future work, we have a plan to apply this method to System Dynamics simulation area. The System Dynamics calculates the performance variables continuously through feedback loops and has different modeling elements compared to this approach. It would be interesting to combine this approach with System Dynamics. Also it would be interesting to explore to integrate PSEE with simulation environment.

REFERENCE

[1] Sutton, S., Heimbigner, D. and Osterweil, L., "Language constructs for managing change in process centered environments" *Fourth SIGSOFT Symposium on Software Development Environments, Software Engineering Notes,* 1990, pp. 206-217.

[2] Kaiser, G.E., Barghouti, N.S. and Sokolsky, M.H., "Preliminary experience with process modeling in the Marvel software development environment kernel" *23d Annual Hawaii International Conference on System Sciences,* IEEE Computer Society, Washington, DC, 1990, pp. 131-140.

[3] Finkelstein, A., Kramer, J. and Nuseibeh, B., *Software Process Modeling and Technology,* Research Studies Press, Somerset, England, 1994.

[4] Min, S., Bae, D., "MAM nets : A Petri-net based Approach to Software Process Modeling, Analysis and Management," *Proc. of Int'l Conf. of Software Engineering and Knowledge Engineering(SEKE'97),* pp. 76-87, June 1997.

[5] Kellner, M.I., Madachy, R., and Raffo, D., "Software process modeling and simulation: Why, What, How?," *Journal of Systems and Software,* Elsevier, 1999, pp. 91-105.

[6] *Software Process Engineering Metamodel Specification, Version 1.1,* OMG Document formal/05-01-06, 2005.

[7] Choi, K., Bae, D.,Kim, T., "DEVS-based software process simulation modeling: Formally specified, modularized, and extensible SPSM," *The 6th International Workshop on Software Process Simulation and Modeling, in conjunction with the 27th ICSE,* IEEE Computer Society Press, Washington, DC, 2005, pp. 73-82.

[8] Acuna, S.T., Juristo, N., *Software Process Modeling,* Springer, 2005.

[9] Zeigler, B.,Pracehofer, H., Kim, T., *Theory of Modeling and Simulation,* Second Edition, Academic Press, New York, 2000.

[10] Kim, T., *DEVSimHLA v2.2.0 Developer's Manual,* Korea Advanced Institute of Science and Technology (KAIST), 2004.

[11] OMG *Unified Modeling Language Specification, Version 1.4,* 2001.

[12] *Apache XML Project Web site,* http://xml.apache.org, 2005.

[13] Kellner, M.I., "Software process modeling example problem" *6th International Software Process Workshop,* IEEE Computer Society, Washington, DC, 1991.

[14] *Enterprise Architect Web site,* http://www.sparxsystem.com.au, Sparx Systems, Inc., 2005.

[15] Kellner, M.I., "Software process modeling support for management planning and control," *First International Conference on the Software Process,* IEEE Computer Society Press, Washington, DC, 1991, pp. 8-28.

박 승 훈

- 2003년 부산대학교 정보컴퓨터공학부(공학사)
- 2003년~현재 한국과학기술원 전산학과(석박사 통합과정)
- 관심분야 : 소프트웨어 프로세스 모델링, 소프트웨어 프로세스 관리, 소프트웨어 프로세스 시뮬레이션 등

최 경 식

- 1992년 육군사관학교 물리학과(이학사)
- 1996년~1998년 Oregon State Univ., Dept. of EECS (공학석사)
- 2003년~2007년 한국과학기술원 전산학과(공학박사)
- 2007년~현재 육군 소령
- 관심분야 : 소프트웨어 프로세스 시뮬레이션, 소프트웨어 프로젝트 발주 관리 등

배 두 환

- 1980년 서울대학교(공학사)
- 1985년~1987년 University of Wisconsin-Milwaukee (공학석사)
- 1988년~1992년 University of Florida (공학박사)
- 1992년~1994년 University of Florida 조교수
- 1995년~현재 한국과학기술원 전산학과 교수
- 관심분야 : 객체 지향 기술, 분산/병렬 소프트웨어 공학, 소프트웨어 개발 방법론/프로세스 모델, 컴포넌트 기반 소프트웨어 공학, 소프트웨어 프로세스 등