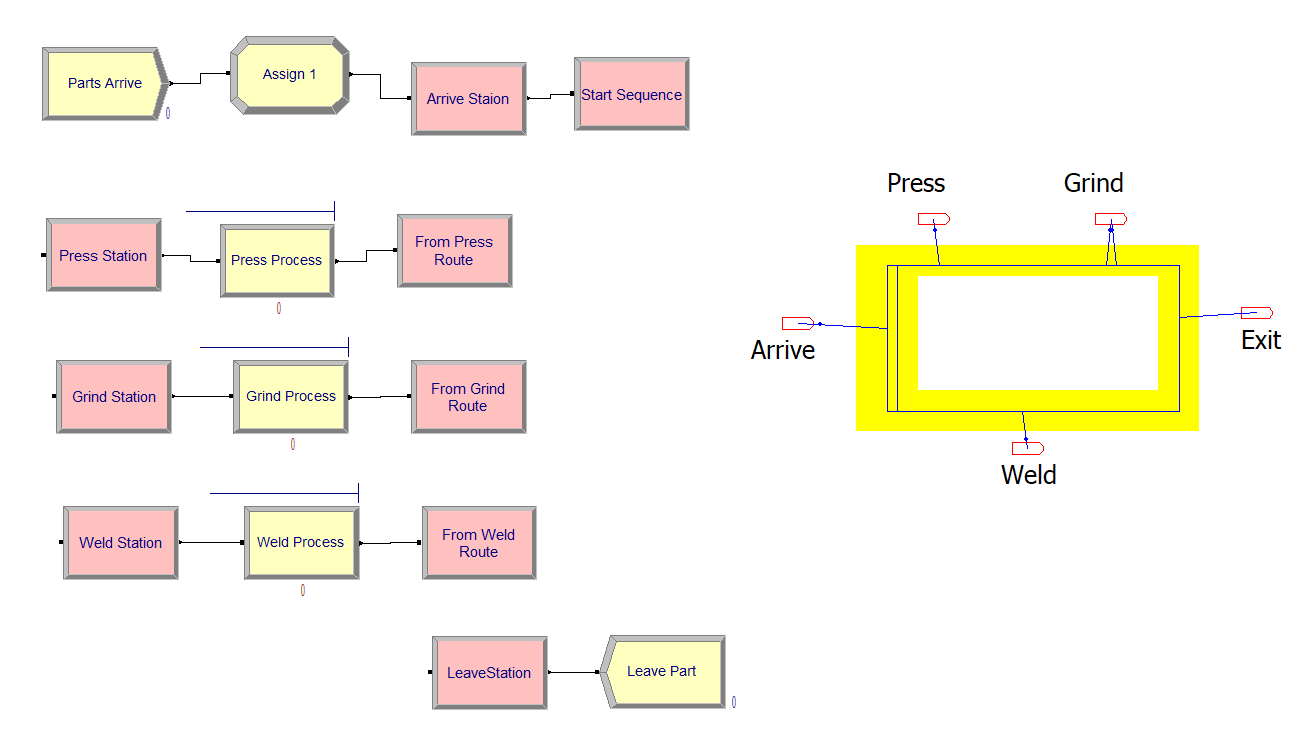
**<1>**

문제 상황에 맞게 아래와 같이 구성했다.



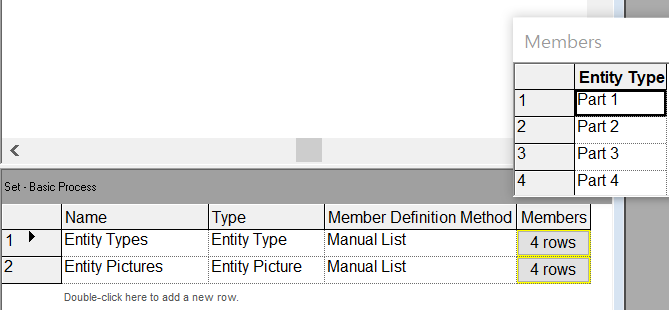
진행한 순서대로 설명하자면, 우선 문제의 조건대로 Parts Arrive 모듈과 Assign 모듈을 추가했다.

Assign 모듈에서는 Entity Type, Entity parts, Entity Pictures, Entity Sequences를 할당했다.

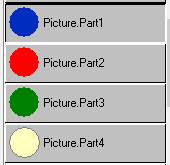
테이블이(가) 표시된 사진

자동 생성된 설명

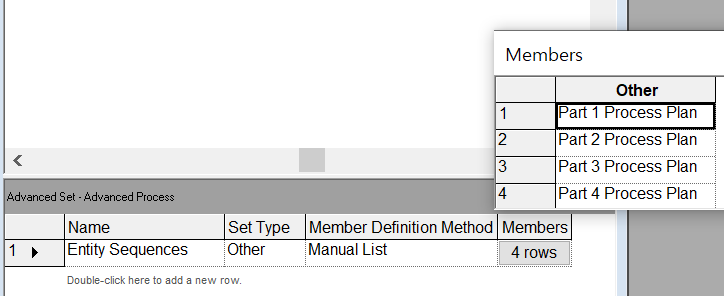
* Entity Types와 Entity Pictures에 해당하는 Set을 만들어 assign 했다.



이 때, Entity pictures는 picture.Ball 4개의 그림을 복제해 사용했다.



* Entity Sequences는 해당하는 AdvancedSet을 만들어 assign 했다.

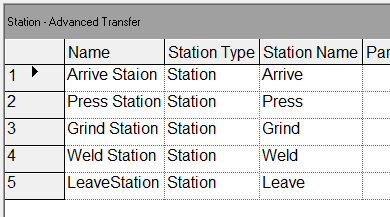


이 때, 각 part의 process plan은 아래와 같이 sequence 모듈에서 설정했다.

테이블이(가) 표시된 사진

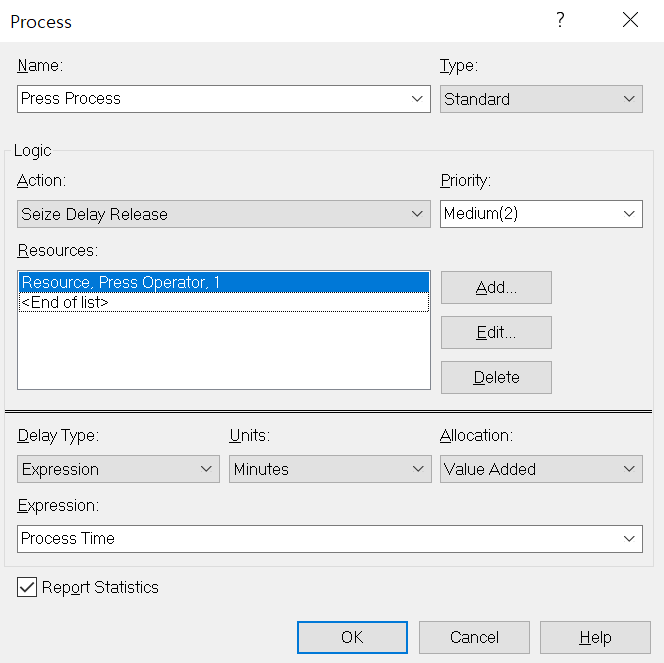
자동 생성된 설명

이후 arrive와 leave를 포함하여 station을 만들어주었다.



각각의 route는 3분의 소요 시간을 고려하고, destination type을 by sequence로 설정했다.

그 후, process 모듈을 적용하기 전에, press / grind/ weld에 각각 1/1/2명의 operator을 resource로서 추가하고, process time만큼 소요되는 seize-delay-release process 모듈을 각 station과 route 사이에 추가했다.



제시해준 대로 10000분의 실행 결과는 다음과 같다.

*“Flowtime, waiting at stations, and use of machines are the important considerations.”*

위 세 부분을 살펴보겠다.

테이블이(가) 표시된 사진

자동 생성된 설명

Total Flowtime은 위와 같다.

테이블이(가) 표시된 사진

자동 생성된 설명

Wait time이 flow time의 상당 부분을 차지하고 있음을 확인할 수 있다.

Waiting at station은 아래와 같다.

테이블이(가) 표시된 사진

자동 생성된 설명

Weld station에서 가장 긴 waiting이 일어나고 있음을 알 수 있다.

Use of machines는 아래와 같다.

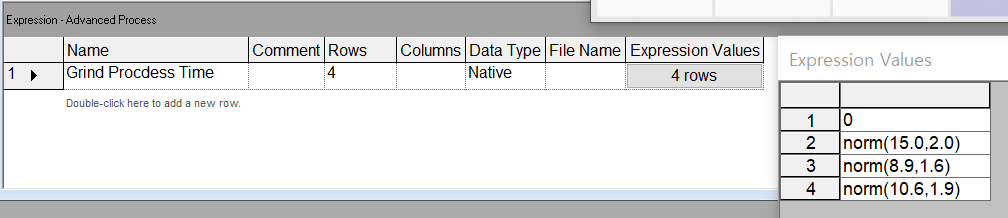
테이블이(가) 표시된 사진

자동 생성된 설명

Weld operator가 가장 오래 사용되고 있음을 확인할 수 있는데, 앞서 분석한 바를 생각했을 때 상식적인 결과이다.

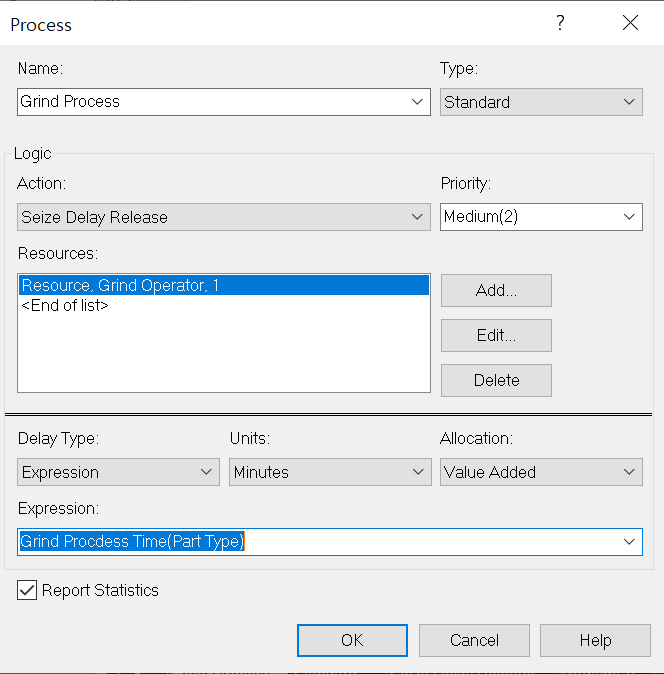
**<2>**

Grind Process Time을 Expression에서 따로 지정해주었다.



Entity Type 순서대로 입력하였다.

이후에, Ground Process 모듈에서 기존의 Process Time이 아닌 Grind Process Time을 입력하였다.



그 결과를 살펴보면 아래와 같다.

마찬가지로 Flowtime, waiting at stations, and use of machines 세 부분을 집중적으로 살펴보겠다.

테이블이(가) 표시된 사진

자동 생성된 설명

아래 기존 flow time과 비교하여, 1회 반복으로 인하여 half width가 제시되지 않아 평균값의 유의미한 차이를 분석하긴 어렵지만, Minimum 값과 maximum 값을 보면 꽤나 차이나는 것을 확인할 수 있다.

테이블이(가) 표시된 사진

자동 생성된 설명

[기존 flowtime]

테이블이(가) 표시된 사진

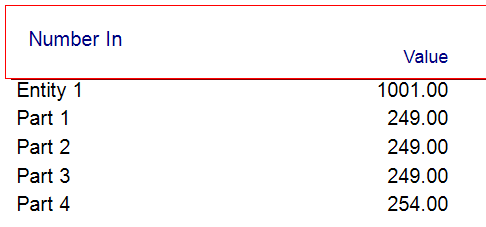
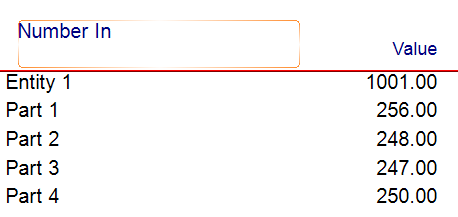
자동 생성된 설명

waiting at stations 역시 min/max 값을 비교해보았을 때 변동성이 있는 모습을 보여주었다. use of machines에서는 가질 수 있는 값의 특성 상 변동성을 직접 확인할 수 없었다.

테이블이(가) 표시된 사진

자동 생성된 설명

이 이유에 대해 생각해보면, 여러 확률변수에 의한 랜덤 값이 달라졌다.

[새로운 Number In] [기존 Number In]

우선, 25%씩의 확률에 의해 결정되는 part 별 number In이 달라졌고, 추가로 normal 분포를 따르는 각 process의 작업 시간이 달라졌을 것이다.

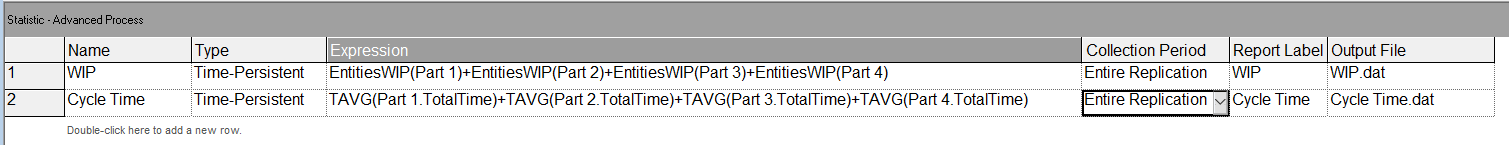
이러한 **확률변수의 변동성**은 초기 상태에서의 큰 차이를 불러일으킬 수 있다.

즉, 시스템이 **안정화 단계에 진입하기 이전**의 상태이기에, 특정 entity의 차이가 큰 영향을 미칠 수 있는 단계이다. 안정화 단계에 진입하기 이전이라는 점은 다음 문항에서 자세히 다루겠다.

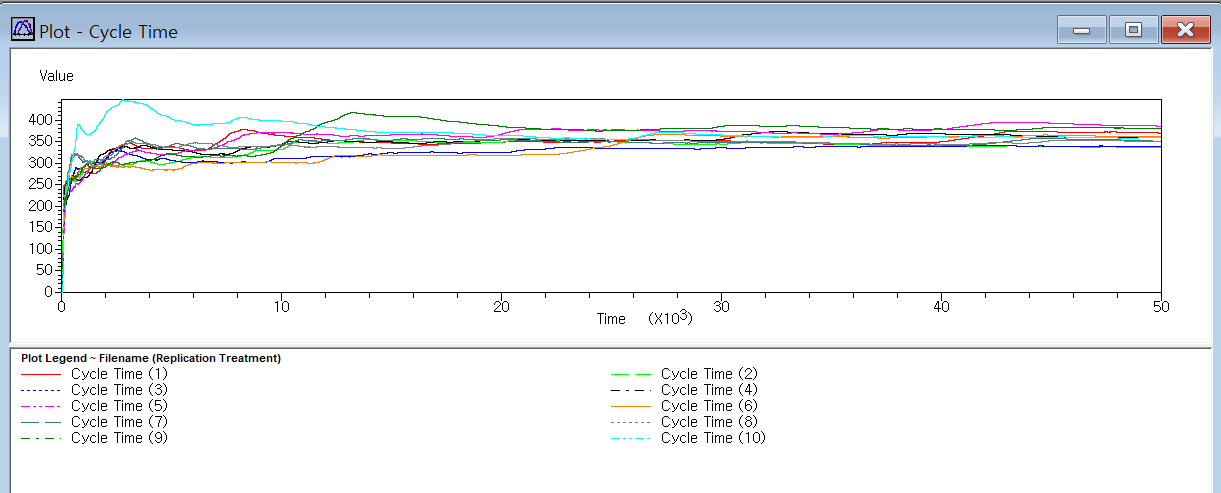
위의 이유들로 인해 <1>번과 <2>번의 결과는 크게 달라졌음을 짐작할 수 있다.

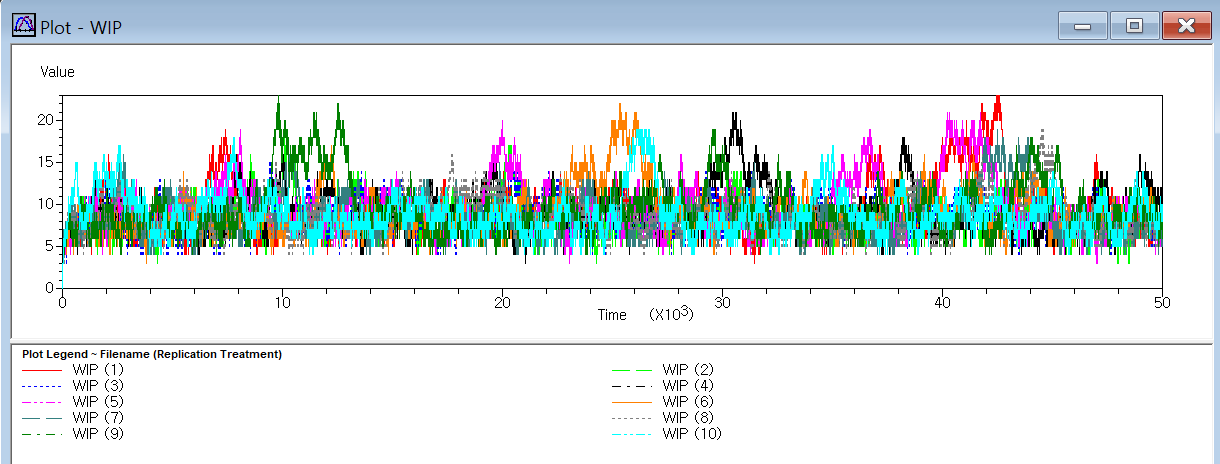
**<3>**

문제에서 제시한 대로 WIP과 Cycle Time을 Statistic으로 설정하고, 각각 dat파일을 생성하여 Output Analyzer을 실행시켰다.



50000분의 10회 Replication 결과를 Plot에 그려보면, 아래와 같다.



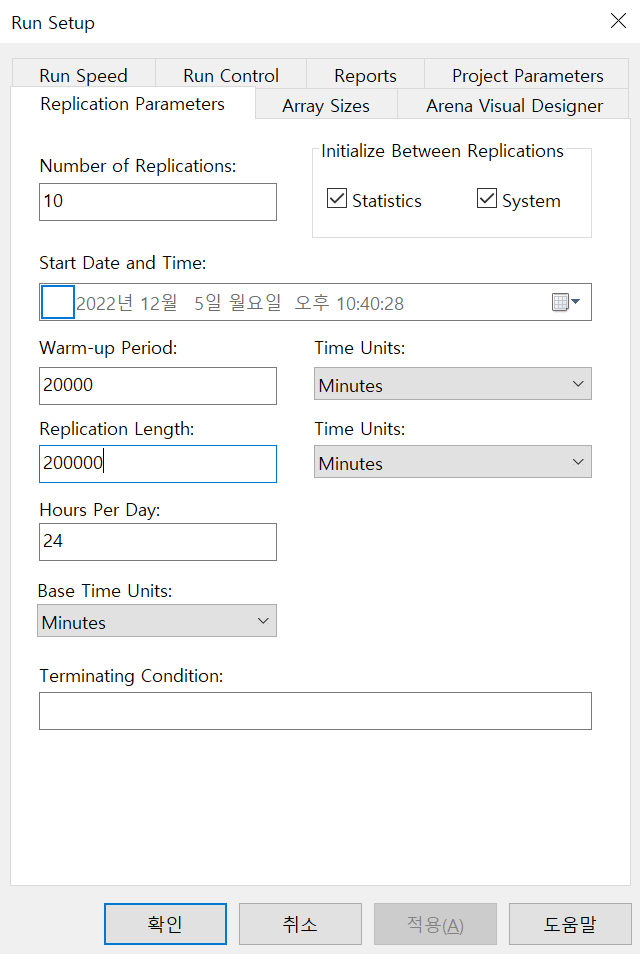


두 그래프로부터, 약 20000 min 이후(보수적 접근)에 안정화되는 것을 확인할 수 있다.

이에 20000min을 warmup period로 설정하였다.

**<4>**

3번에서 설정한 20000min이 run length의 10%라고 가정하자 했으므로, 이에 맞추어 모델을 설정해주었다.



average WIP의 95% 신뢰구간과, 1번에서의 average WIP의 95% 신뢰구간을 구하면 아래와 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

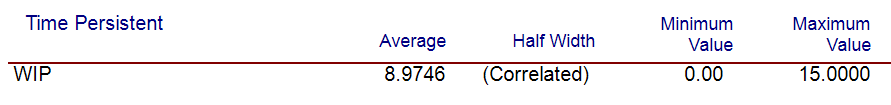
텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

[4번 average WIP의 95% 신뢰구간]

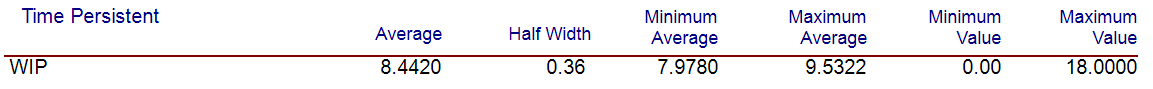


테이블이(가) 표시된 사진

자동 생성된 설명

[1번 average WIP의 95% 신뢰구간]

1번 문제의 신뢰구간을 구해보면, 1회의 replication만을 적용하기에 95% 신뢰구간이 대부분 구해지지 않는다. 다만, part3의 신뢰구간은 4번에서 약 1/10 수준으로 줄어든 것을 확인할 수 있다. 이는 replication의 차이와 run length의 차이에 의한 것으로 생각된다.



테이블이(가) 표시된 사진

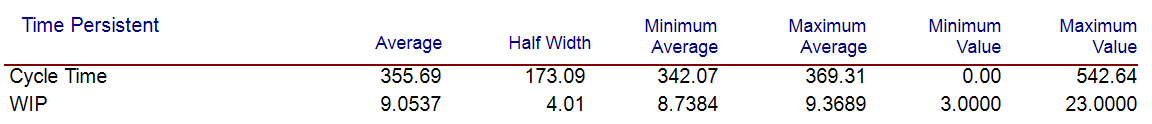
자동 생성된 설명

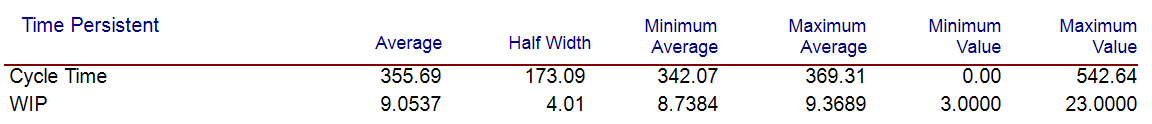
[1번(10 replications) average WIP의 95% 신뢰구간]

1번 문제의 상황을 4번 문제와 같이 10회의 replications를 진행하더라도, run length의 차이에 의해 95% 신뢰구간이 2배 이상 넓게 나타나는 것을 확인할 수 있다.

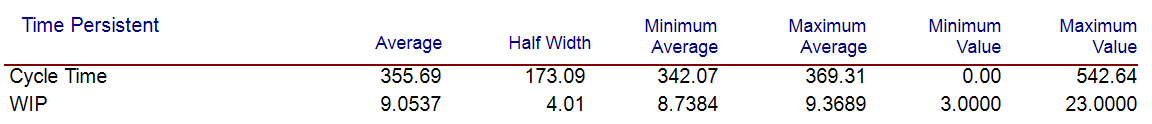
**<5>**

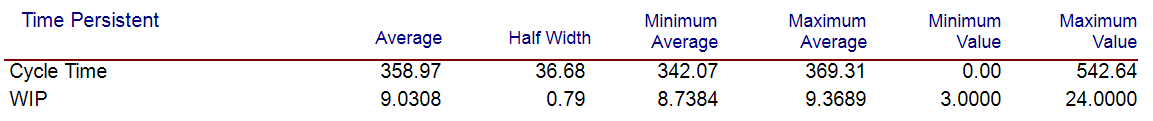
average overall WIP의 95% 신뢰구간이 평균의 ± 5%가 되기 위해서 몇 번의 Repilcations가 필요한지 시행착오법을 통해 알아보았다.



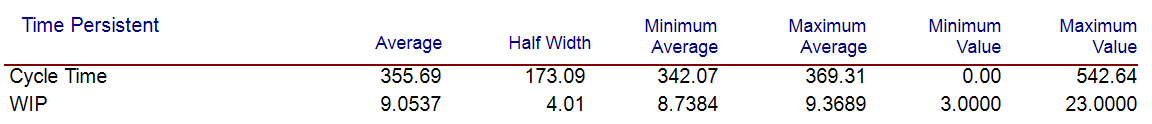


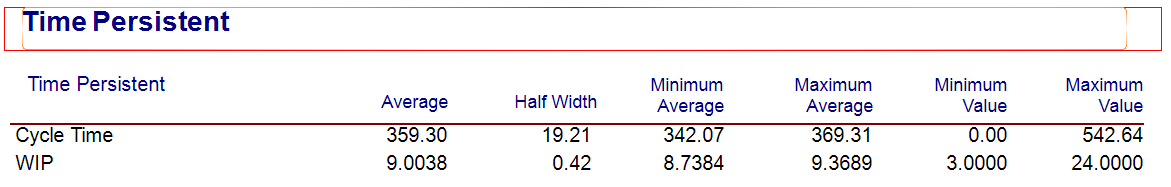
[2 Replications : ± 45%]





[3 Replications : ± 8.75%]



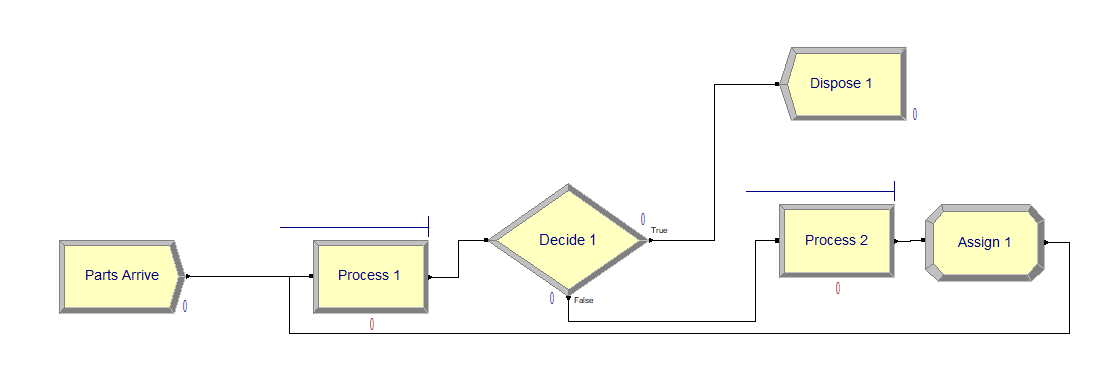


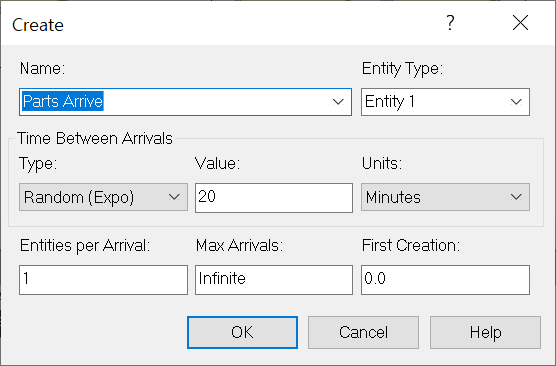
[4 Replications : ± 4.66%]

즉, average overall WIP의 95% 신뢰구간이 평균의 ± 5%가 되기 위해서 4번의 Repilcations가 필요하다.

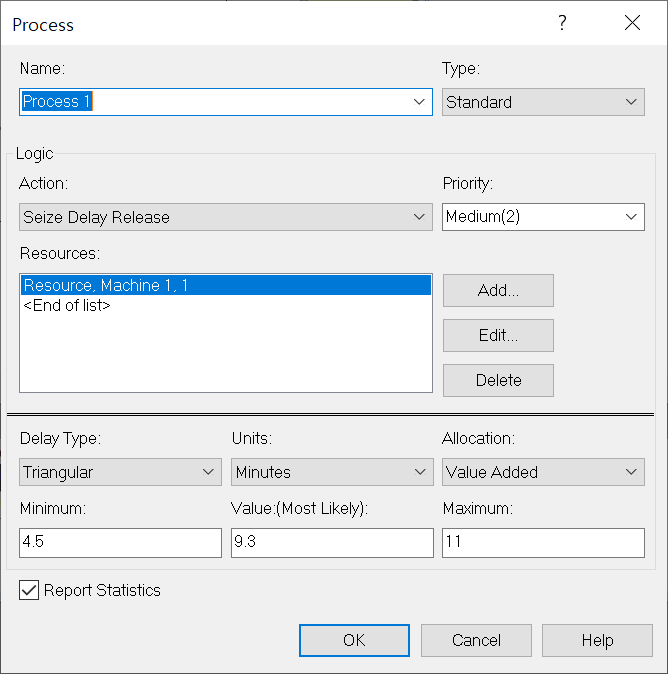
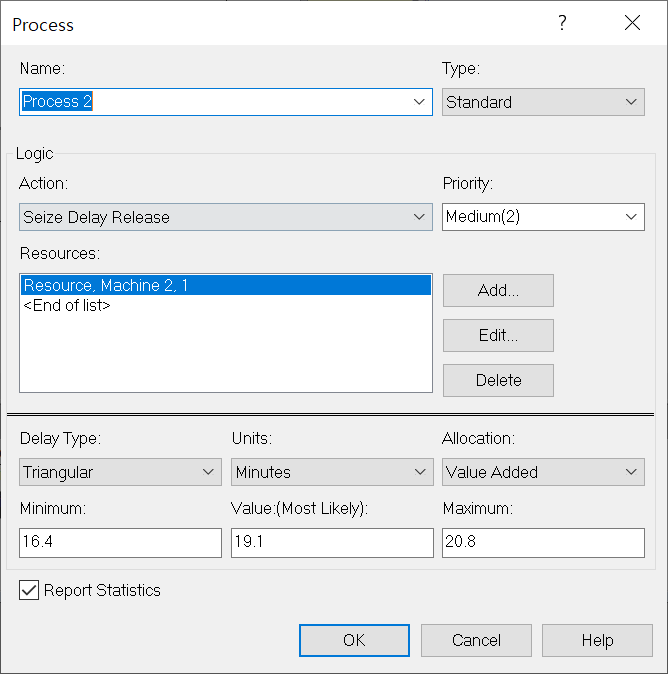
**<연습문제 8-3>**

연습문제 5-2를 변경하는 문제이므로, 우선 5-2를 모델링했다.





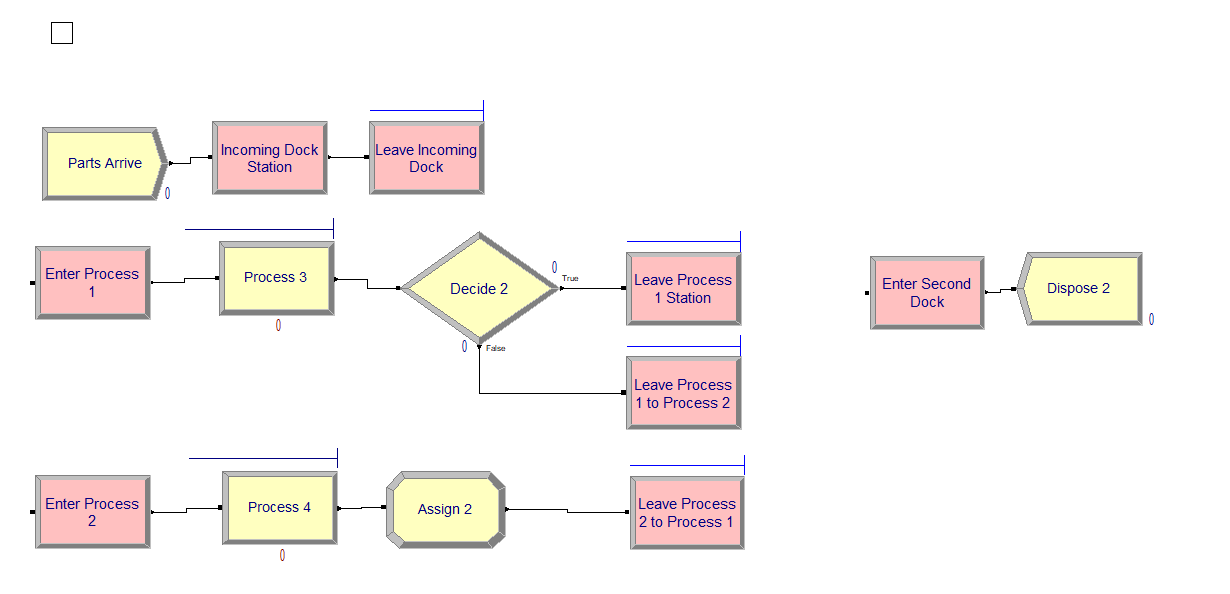
Machine 1, 2를 resource로 정의해주고, 아래와 같이 process 1, 2를 설정했다.

Process 1을 거친 후에는 process 2 done이라는 attribute 값이 2인지 확인하고, 2라면 dispose를, 아니라면(초기값은 2가 아님) process 2로 넘어간다.

Process 2를 한 번 거친 entity는 process 2 done이라는 attribute에 2라는 값을 assign하여, process 1로 돌아가도록 해 문제의 상황대로 시뮬레이션을 구현하였다.

이 5-2를 8-3의 상황에 맞게 시뮬레이션했다.

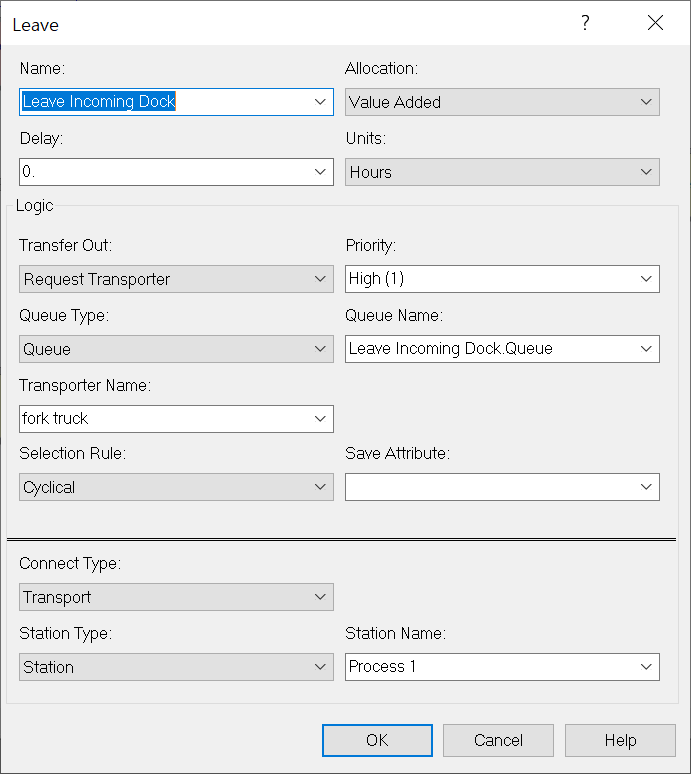
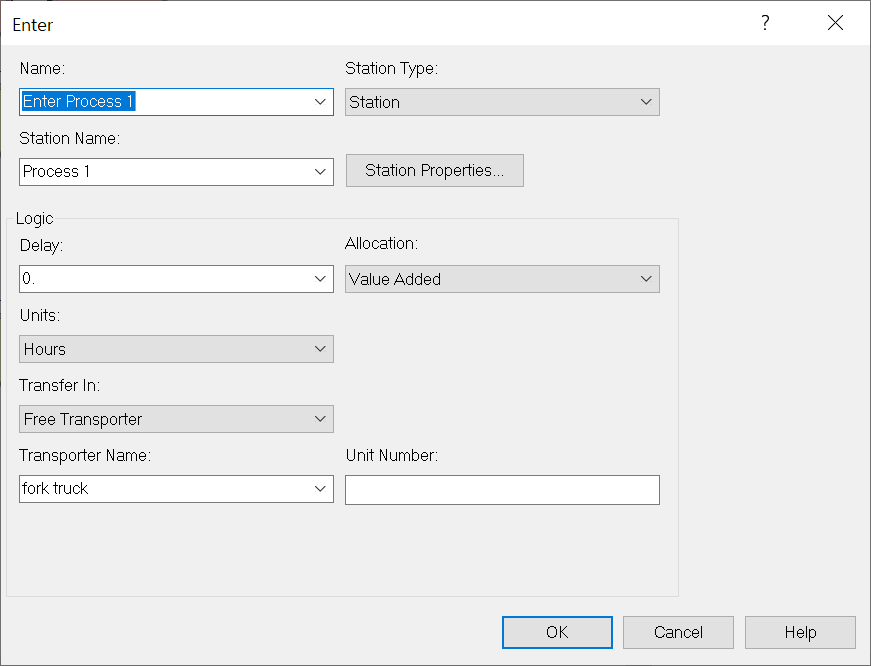


Incoming Dock, Process 1, Process 2, Second Dock station을 만든 후, 5-2의 상황을 Leave 모듈과 Enter 모듈을 활용해 다시 표현하였다.

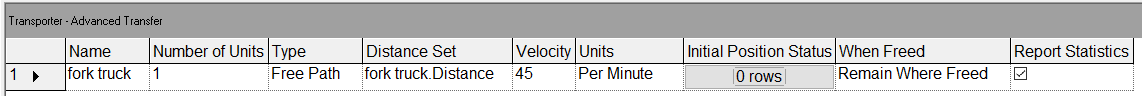
테이블이(가) 표시된 사진

자동 생성된 설명

세부 설정은 Leave Incoming dock, enter process 1만 예시로 보자면 아래와 같다.

또한, fork truck transporter을 설정해주었다.



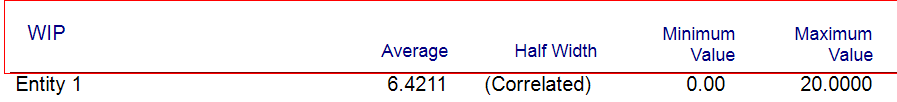
이 때, distance를 설정해주었는데, 실제로 entity가 움직이는 경로와 transporter의 이동을 위한 경로를 전부 지정해주었다.

테이블이(가) 표시된 사진

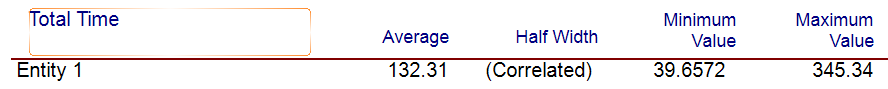
자동 생성된 설명

이렇게 구현한 시뮬레이션을 문제에서 제시한 대로 20000 min, 1 replication으로 실행했다.

average number in the machine queues : 6.42개

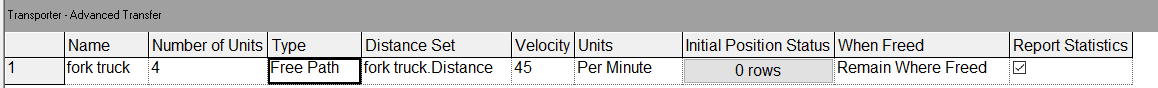


the average part cycle time : 132.31분



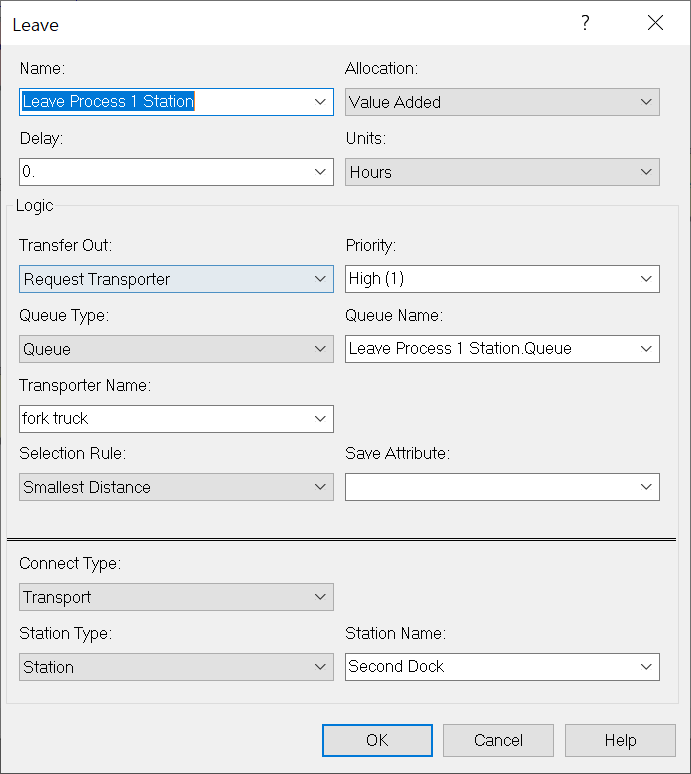
**<연습문제 8-4>**

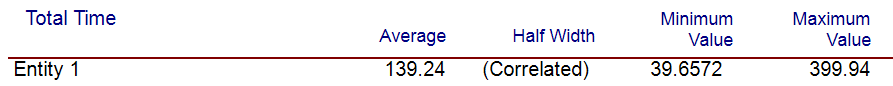
우선, 문제의 상황대로 the number of transporters을 4로 설정했다.



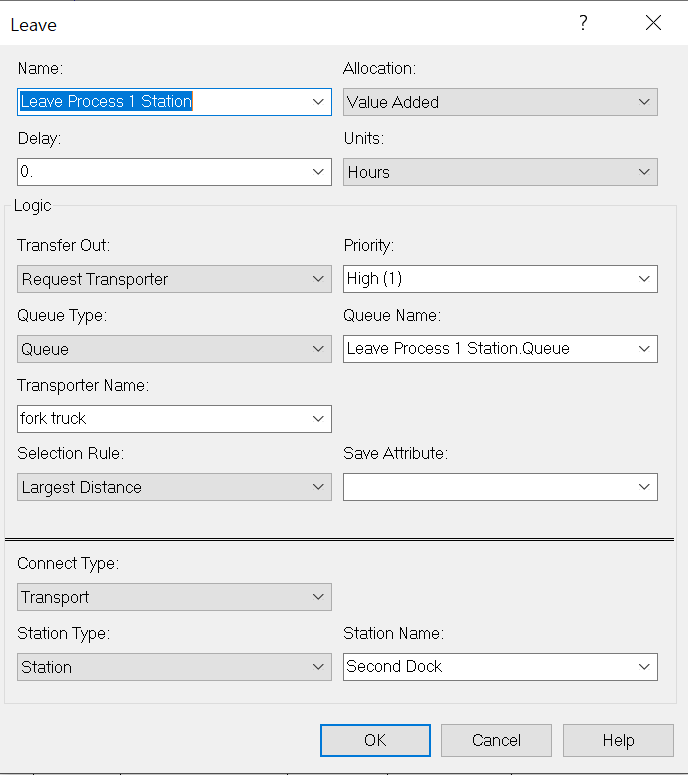
이후 transporter selection rules를 Smallest Distance / Largest Distance / Cyclical으로 바꾸어가며 모델을 실행(20000 min)하여 average cycle time을 비교해보았다.

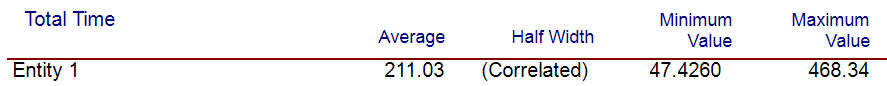
Smallest Distance : **139.24 min**





Largest Distance : **211.03 min**





Cyclical : **136.37 min**

