(12) 특허협력조약에 의하여 공개된 국제출원

(19) 세계지식재산권기구
국제사무국

(43) 국제공개일
2012 년 6 월 28 일 (28.06.2012)

WIPO | PCT

(10) 국제공개번호
WO 2012/086948 A2

(51) 국제특허분류:
*G06F 9/44* (2006.01)

(21) 국제출원번호: PCT/KR2011/009405

(22) 국제출원일: 2011 년 12 월 7 일 (07.12.2011)

(25) 출원언어: 한국어

(26) 공개언어: 한국어

(30) 우선권정보:
10-2010-0132017 2010 년 12 월 21 일 (21.12.2010) KR

(71) 출원인 (US 을(를) 제외한 모든 지정국에 대하여): **현대중공업 주식회사 (HYUNDAI HEAVY INDUSTRIES CO., LTD.)** [KR/KR]; 울산광역시 동구 방어진 순환도로 1000, 682-792 Ulsan (KR).

(72) 발명자; 겸
(75) 발명자/출원인 (US 에 한하여): **강호열 (KANG, Ho Yeol)** [KR/KR]; 울산광역시 남구 야음동 쌍용스윗닷홈 105 동 403 호, 680-040 Ulsan (KR).

(74) 대리인: **이익상 (LEE, Ik Sang)**; 서울특별시 강남구 역삼동 783-43 삼화빌딩 3 층, 135-080 Seoul (KR).

(81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

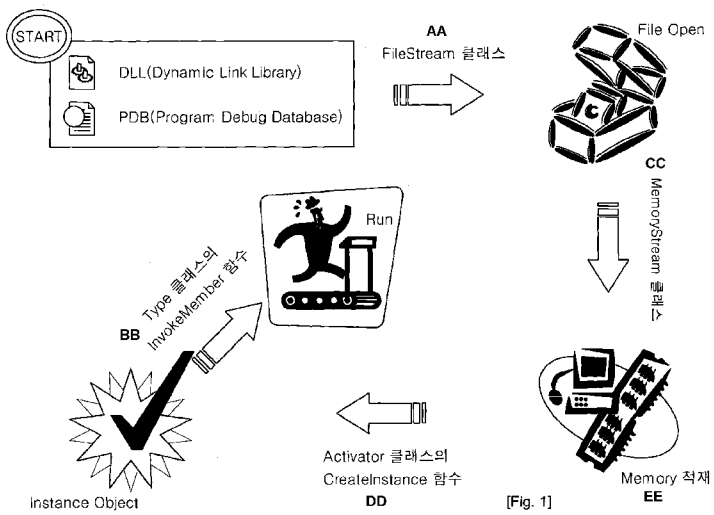
(84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), 유럽 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

공개:
— 국제조사보고서 없이 공개하며 보고서 접수 후 이를 별도 공개함 (규칙 48.2(g))

WO 2012/086948 A2

(54) **Title**: METHOD FOR USING REAL-TIME LOADING OF DLL FOR AM CAD

(54) **발명의 명칭** : 에이엠 캐드용 디엘엘의 실시간 로딩 사용방법




[Fig. 1]

AA ... FileStream Class
BB ... InvokeMember function of Type Class
CC ... MemoryStream Class
DD ... CreateInstance function of Activator Class
EE ... Memory loading

(57) **Abstract**: The present invention relates to a method for using real-time loading of DLL for AM CAD. Said method is for improving the effectiveness over time of the use of AM CAD by loading a DLL file onto memory in real-time and then making same available for use right after the loading without a the rebooting of the computer or program. To this end, the present invention includes: step 1 of opening a DLL file for the AM CAD program in a file stream format and outputting it in a stream format; step 2 of stacking the DLL file in the stream format onto memory; step 3 of converting the DLL file in the stream format, which is mounted onto memory, into an instance object format available in the AM CAD program; and step 4 of dynamically calling the instance object.

(57) **요약서**: 본 발명은 AM 캐드용 DLL 의 실시간 로딩 사용방법에 관한 것으로서, 이는 컴퓨터, 프로그램 등의 재부팅 과정을 거치지 않고 실시간으로 DLL 파일을 메모리상에 로딩시킨 다음 바로 사용가능하도록 하여 AM 캐드 사용의 시간 대비 효율성을 향상시키기 위한 것이다. 이를 위해 본 발명은, AM 캐드 프로그램용 DLL 파일을 파일스트림 형태로 오픈하여 스트림 형태로 출력하는 제 1 단계와; 상기 스트림 형태의 DLL 파일을 메모리상에 적재하는 제 2 단계와; 메모리상에 적재된 상기 스트림 형태의 DLL 파일을 AM 캐드 프로그램에서 사용가능한 인스턴스 오브젝트 형태로 변환하는 제 3 단계와; 상기 인스턴스 오브젝트를 동적으로 호출하는 제 4 단계;로 이루어지는 것을 특징으로 한다.

# 명세서

# 발명의 명칭: 에이엠 캐드용 디엘엘의 실시간 로딩 사용방법

## 기술분야

[1] 본 발명은 AM 캐드용 DLL의 실시간 로딩 사용방법에 관한 것으로서, 보다 구체적으로는 컴퓨터, 프로그램 등의 재부팅 과정을 거치지 않고 실시간으로 DLL 파일을 메모리상에 로딩시킨 다음 바로 사용가능하도록 하여 AM 캐드 사용의 시간대비 효율성을 향상시키기 위한 AM 캐드용 DLL의 실시간 로딩 사용방법에 관한 것이다.

[2]

## 배경기술

[3] 일반적으로 프로그램 작성 시 소스 코드를 컴파일 후 목적파일(*.obj)이 만들어지면 이들을 서로 연결하는 링크 과정으로 들어가게 되고, 여기서의 링크는 컴파일로 만들어진 여러 목적파일을 묶어 하나의 *.exe파일로 출력하게 되며, 이렇게 컴파일 과정에서 링크되는 것을 정적 링크(Static Link)라고 한다. 정적 링크를 사용하면 하나의 실행파일에 대해서만 유효하다. 즉, 실행파일이 여러개 있을때 똑같은 기능을 하는 함수가 필요하다면 각 실행파일에 같은 함수들이 모두 정의되어야하기 때문에 실행시 많은 코드들이 메모리에 로드되어 필요없는 공간을 사용하게 되는 단점을 갖는다.

[4] 기술된 정적링크와 대비되는 의미의 동적링크(Dynamic Link)는 링크 과정에서 묶이는 것이 아니라 실행시간(Run-Time)에 연결되는 코드라고 정의할 수 있다. 즉, 실행파일과는 별도로 존재하고 실행파일이 실행되면서 동적으로 링크되어 사용된다. 동적링크를 사용하면 많은 실행파일들이 메모리에 로드된 하나의 함수를 공유할수 있기 때문에 메모리나 하드 디스크의 낭비를 줄일 수 있고 프로그램의 실행속도가 빨라질 수 있다.

[5] 이러한 동적링크의 장점을 활용한 DLL(Dynamic Link Library)의 경우 다른 프로그램으로 인해 이미 메모리에 로드된 상태라면 로드과정이 필요없어지게 되지만, 현재까지 공지된 기술에 따르면 컴퓨터, 프로그램 등의 재부팅 과정을 거치지 않고 실시간으로 새로운 DLL 파일을 메모리상에 로딩시키기 어려운 문제점이 있다.

[6]

## 발명의 상세한 설명

## 기술적 과제

[7] 본 발명은 상기한 종래 기술의 문제점을 해결하기 위해 안출된 것으로서, 컴퓨터, 프로그램 등의 재부팅 과정을 거치지 않고 실시간으로 DLL 파일을 메모리상에 로딩시킨 다음 바로 사용가능하도록 하여 AM 캐드 사용 시간대비 효율성을 향상시킬 수 있는 AM 캐드용 DLL의 실시간 로딩 사용방법을

제공하는데 그 목적이 있다.

[8]

## 과제 해결 수단

[9] 상기 목적을 달성하기 위하여, 본 발명은, AM 캐드 프로그램용 DLL 파일을 파일스트림 형태로 오픈하여 스트림 형태로 출력하는 제1 단계와; 상기 스트림 형태의 DLL 파일을 메모리상에 적재하는 제2 단계와; 메모리상에 적재된 상기 스트림 형태의 DLL 파일을 AM 캐드 프로그램에서 사용가능한 인스턴스 오브젝트 형태로 변환하는 제3 단계와; 상기 인스턴스 오브젝트를 동적으로 호출하는 제4 단계;로 이루어지는 것을 특징으로 한다.

[10] 상기 제1 단계에서는 DLL 파일과 함께 PDB(Program Debug Database) 파일을 오픈하여 스트림 형태로 출력하는 것을 특징으로 한다.

[11]

## 발명의 효과

[12] 상술된 바와 같이, 본 발명에 따른 AM 캐드용 DLL의 실시간 로딩 사용방법은 DLL 파일을 스트림 형태로 메모리에 적재한 상태에서 AM 캐드 프로그램에서 사용가능한 인스턴스 오브젝트 형태로 변환하여 컴퓨터, 프로그램 등의 재부팅 과정을 거치지 않아도 실시간으로 새로운 DLL 파일을 메모리상에 로딩시킨 다음 바로 사용가능하게 한다.

[13] 또한 DLL 파일을 스트림 형태로 메모리에 적재할 시 PDB 파일 또한 함께 전송하여 DLL 코드의 오류에 대한 확인작업이 보다 용이하게 진행되도록 할 수 있다.

[14]

## 도면의 간단한 설명

[15] 도 1은 본 발명에 따른 AM 캐드용 DLL의 실시간 로딩 사용방법을 개략적으로 도시한 도면.

[16]

## 발명의 실시를 위한 최선의 형태

[17] 이하, 도면을 참조로 하여 본 발명에 따른 AM 캐드용 DLL의 실시간 로딩 사용방법을 설명하기로 한다.

[18]

[19] 도 1은 본 발명에 따른 AM 캐드용 DLL의 실시간 로딩 사용방법을 개략적으로 도시한 도면이다.

[20]

[21] 본 발명에 따른 AM 캐드용 DLL의 실시간 로딩 사용방법은 기본적으로 AM 캐드 프로그램용 DLL 파일을 파일스트림 형태로 오픈하여 스트림 형태로 출력하는 제1 단계와, 상기 스트림 형태의 DLL 파일을 메모리상에 적재하는 제2 단계와, 메모리상에 적재된 상기 스트림 형태의 DLL 파일을 AM 캐드

프로그램에서 사용가능한 인스턴스 오브젝트 형태로 변환하는 제3 단계와, 상기 인스턴스 오브젝트를 동적으로 호출하는 제4 단계로 이루어진다.

[22]

[23] 상기 제1 및 제2 단계에서는 FileStream 클래스(파일을 읽고 쓰는 기능을 제공)를 통해 DLL 파일을 오픈하여 스트림 형태로 출력한 다음, 메모리 스트림 객체 및 버퍼를 생성하여 DLL 파일 스트림이 메모리상에 저장되도록 하며, 이의 프로그램 소스코드 에는 다음과 같다.

[24]

```
[25] Assembly assembly = null;
[26] …
[27] using (FileStream fs = File.Open(dllName, FileMode.Open))
[28] {
[29] using (MemoryStream ms = new MemoryStream())
[30] {
[31] byte[] byteBuffer = new byte[1024];
[32] int nRead = 0;
[33] while ((nRead = fs.Read(byteBuffer, 0, 1024)) > 0)
[34] {
[35] ms.Write(byteBuffer, 0, nRead);
[36] }
[37] assembly = Assembly.Load(ms.ToArray());
[38] }
[39] }
```

[40]

[41] 상기 소스코드에 대해 구체적으로 살펴보면, Assembly형 객체 Assembly를 생성하고, DLL 파일을 열어서 FileStream형 객체 fs에 스트림형태로 받아온 다음, 메모리 스트림 객체를 생성한다. 그 후 1024바이트 크기의 버퍼를 생성하고, Fs 스트림의 끝까지 1024 바이트씩 읽어들여 Ms에 기록한다. Ms에 기록된 데이터를 Assembly 클래스형에 맞게 assembly 객체에 저장한다.

[42]

[43] 이와 같은 상기 소스코드의 과정에 따라 DLL 파일을 메모리에 적재해 놓았다가 함수형태로 호출하여 사용할 수 있게 되며, 참고로 MemoryStream 클래스에서는 디스크나 네트워크 연결 대신 메모리를 백업 저장소로 사용하는 스트림을 만들고, MemoryStream 객체 작성 시 초기화되거나 비어 있게 만들어진 부호 없는 바이트 배열로 저장된 데이터를 캡슐화하며, 캡슐화된 데이터는 메모리에서 직접 액세스할 수 있음에 따라 응용 프로그램의 임시 버퍼 및 파일의 필요성을 감소시키는 기능을 수행한다.

[44]

[45] 상기 제2 내지 제4 단계에서는 메모리에 저장된 DLL 파일 스트림을 AM 캐드 프로그램에서 사용가능한 인스턴스 오브젝트 형태로 변환하여 동적 출력가능한 형태로 가공하며, 이의 프로그램 소스코드 예는 다음과 같다.

[46]

[47] foreach (Type type in assembly.GetExportedTypes())

[48] {

[49] if (type.FullName.Equals(typeName))

[50] {

[51] object[] argsConstructor = new object[] { };

[52] object instance = Activator.CreateInstance(type,

[53] argsConstructor);

[54] object[] argsMethod = new object[] { };

[55] object outValue = type.InvokeMember(methodName,

[56] BindingFlags.Default | BindingFlags.InvokeMethod,

[57] null, instance, argsMethod);

[58] }

[59] }

[60]

[61] 상기 소스코드에 대해 구체적으로 살펴보면, assembly 객체에 저장된 데이터 타입 중에 Type형 type과 같은 것들만 추출하고, 타입명(typeName)과 같으면 새로운 오브젝트를 argsConstructor이라는 배열형태 이름으로 생성하며, DLL 안에 methodName 함수를 호출하기 위해서 InvokeMember를 사용한다. 여기서 InvokeMember 안의 argument들은 단순히 그 InvokeMember 함수를 사용하기 위해 기록된 것이다.

[62]

[63] 참고로 Activator 클래스의 CreateInstance 함수는 지정된 매개 변수와 가장 일치하는 생성자를 사용하여 지정된 형식의 인스턴스를 만들게 된다.

[64] 한편으로 상기 제1 단계에서는 DLL 파일과 함께 PDB(Program Debug Database) 파일을 오픈하여 스트림 형태로 출력할 수 있는데, 다시 말해서 DLL 파일을 스트림 형태로 메모리에 적재할 시 PDB 파일 또한 함께 전송하여 DLL 코드의 오류에 대한 확인작업이 보다 용이해지도록 할 수 있으며, 이러한 PDB와 DLL 파일을 동시에 사용하기 위한 프로그램 소스코드의 예는 하기와 같다.

[65]

[66] sing (FileStream fs = File.Open(dllName, FileMode.Open), pfs = File.Open(pdbName, FileMode.Open))

[67] {

[68] using (MemoryStream ms = new MemoryStream(), pms = new MemoryStream())

[69] {

[70] ...

[71] assembly = Assembly.Load(ms.ToArray(), pms.ToArray());

[72] }

[73] }

[74]

[75] 추가로 본 발명에서는 상기 제1 내지 제4 단계를 거친 DLL 및 PDB 파일을 외부단말기에서 호출하는 제5 단계를 더 포함할 수 있는데, 이는 서버상의 메모리에 DLL 파일을 저장하여 네트워크상에서 그 서버와 접속된 복수개의 외부단말기가 실시간으로 그 DLL 파일을 AD 캐드 프로그램에서 사용가능하도록 하기 위한 것이며, 이를 위한 PML(Product Modeling Language)의 일례는 하기와 같다.

[76] define function !!HHIDllAgent()

[77] import 'HHI.Marine.Hull.Utility.Agent'

[78] handle any

[79] import 'Curved_DLL/HHI.Marine.Hull.Utility.Agent'

[80] handle any

[81] endhandle

[82] endhandle

[83] using namespace 'HHI.Marine.Hull.Utility.Agent'

[84] !dll = '\\10.4.20.34\DLL\HHI.Marine.Field.Any.dll'

[85] !typ = 'HHI.Marine.Any.AnyWrapper'

[86] !exe = 'Show'

[87] !pdb = '\\10.4.20.34\PDB\HHI.Marine.Field.Any.pdb'

[88] !obj = object HHIDllAgentWrapper()

[89] !obj.Execution(!dll, !typ, !exe, !pdb)

[90] endfunction

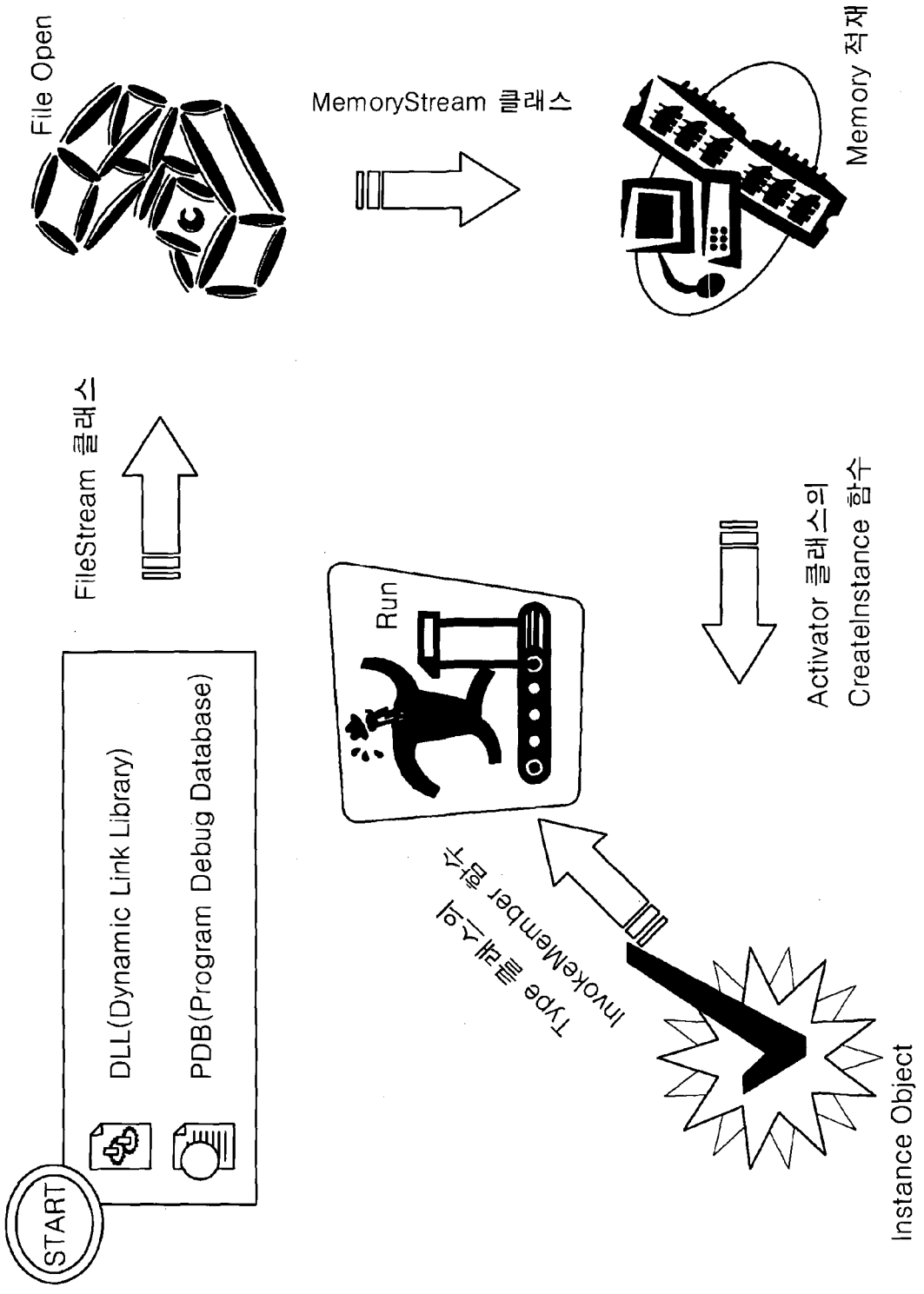
# 청구범위

[청구항 1] AM 캐드 프로그램용 DLL 파일을 파일스트림 형태로 오픈하여 스트림 형태로 출력하는 제1 단계와;
상기 스트림 형태의 DLL 파일을 메모리상에 적재하는 제2 단계와;
메모리상에 적재된 상기 스트림 형태의 DLL 파일을 AM 캐드 프로그램에서 사용가능한 인스턴스 오브젝트 형태로 변환하는 제3 단계와;
상기 인스턴스 오브젝트를 동적으로 호출하는 제4 단계;로 이루어지는 것을 특징으로 하는 AM 캐드용 DLL의 실시간 로딩 사용방법.

[청구항 2] 청구항 1에 있어서,
상기 제1 단계에서는 DLL 파일과 함께 PDB(Program Debug Database) 파일을 오픈하여 스트림 형태로 출력하는 것을 특징으로 하는 AM 캐드용 DLL의 실시간 로딩 사용방법.

[청구항 3] 청구항 2에 있어서,
상기 제1 내지 제4 단계를 거친 DLL 및 PDB 파일을 외부단말기에서 호출하는 제5 단계가 더 포함되는 것을 특징으로 하는 AM 캐드용 DLL의 실시간 로딩 사용방법.

[Fig. 1]


File Open
MemoryStream 클래스
Memory 적재
FileStream 클래스
Activator 클래스의
CreateInstance 함수
Run
DLL(Dynamic Link Library)
PDB(Program Debug Database)
START
Type 클래스의
InvokeMember 함수
Instance Object