

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SAP America, Inc.,
Petitioner,

v.

ISIX IP LLC,
Patent Owner.

IPR2024-00615
U.S. Patent No. 6,308,178

**PETITION FOR *INTER PARTES* REVIEW
UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104**

TABLE OF CONTENTS

I.	INTRODUCTION	6
II.	GROUND FOR STANDING	9
III.	RELIEF REQUESTED AND THE REASONS FOR THE REQUESTED RELIEF	9
IV.	SUMMARY OF THE '178 PATENT	9
V.	PROSECUTION HISTORY	12
VI.	LEVEL OF ORDINARY SKILL IN THE ART	13
VII.	CLAIM CONSTRUCTION	14
VIII.	IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE	14
A.	Challenged Claims	15
B.	Statutory Grounds for Challenges	15
C.	Prior Art Overview	15
1.	Summary of Brann	16
2.	Summary of Suresh	19
D.	Ground 1: Claims 1, 3, 4, 7 and 8 are obvious under 35 U.S.C. §103 over Brann	20
1.	Claim 1	20
2.	Claim 3	36
3.	Claim 4	38
4.	Claim 7	39
5.	Claim 8	40
E.	Grounds 2 and 3: Claims 1 – 5, 7, 9, 10, 23 – 25 are obvious under 35 U.S.C. §103 over Suresh alone or combined with Brann.	41
1.	Reasons to Combine Suresh with Brann	41
2.	Claim 1	44
3.	Claim 2	60
4.	Claim 3	61

5.	Claim 4.....	64
6.	Claim 5.....	66
7.	Claim 7.....	68
8.	Claim 9.....	70
9.	Claim 10.....	70
10.	Claim 23.....	71
11.	Claim 24.....	79
12.	Claim 25.....	79
F.	Grounds 4 and 5: Claims 8 and 19 are obvious under 35 U.S.C. §103(a) over Suresh in view Beauchamp or Suresh in view of Brann and Beauchamp.....	79
1.	Summary of Beauchamp.....	79
2.	Reasons to Combine Suresh with Beauchamp, and Brann	80
3.	Claim 8.....	82
4.	Claim 19.....	83
IX.	DISCRETIONARY DENIAL WOULD BE INAPPROPRIATE	85
A.	Discretionary denial under the <i>Fintiv</i> factors is not appropriate.....	85
1.	No evidence regarding a stay	86
2.	Parallel proceeding trial date	86
3.	Investment in the parallel proceeding.....	86
4.	Overlapping issues with the parallel proceeding	87
5.	Petitioner is a defendant.....	87
6.	Other circumstances.....	87
X.	CONCLUSION.....	88
XI.	MANDATORY NOTICES.....	89
A.	Real Party-in-Interest	89
B.	Related Matters	89
C.	Lead and Back-up Counsel and Service Information	89

CERTIFICATE OF WORD COUNT	91
CERTIFICATE OF SERVICE	92

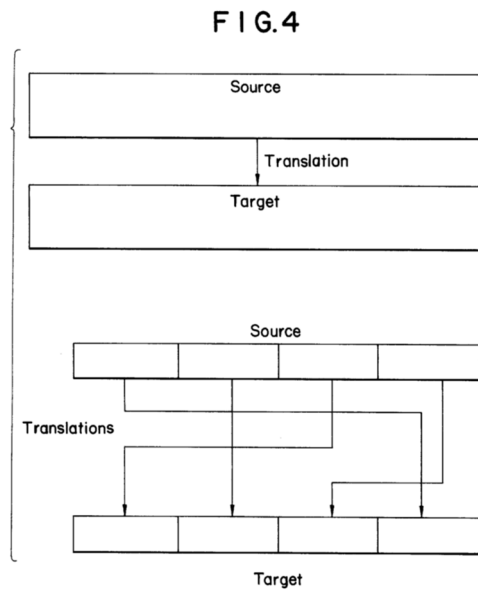
PETITIONER'S EXHIBIT LIST

Ex. 1001	U.S. Patent No. 6,308,178
Ex. 1002	Declaration of Dr. Henry Houh under 37 C.F.R. § 1.68
Ex. 1003	U.S. Patent No. 5,842,205 to Brann
Ex. 1004	U.S. Patent No. 6,208,990 to Suresh et al.
Ex. 1005	Prosecution History of U.S. Patent No. 6,308,178
Ex. 1006	U.S. Patent No. 6,621,505 to Beauchamp et al.
Ex. 1007	Claim Construction Order (ECF No. 37), No. 6:22-cv-201-ADA-DTG, <i>Isix IP LLC v. Int'l Bus. Machines Corp.</i> , Jan. 4, 2023.
Ex. 1008	Patent Owner Isix IP LLC's Preliminary Response (PTAB-IPR2022-01488-6)
Ex. 1009	Excerpts from Microsoft Press Computer Dictionary, Third Edition
Ex. 1010	U.S. Patent 6,014,670 Zamanian et al.
Ex. 1011	U.S. Patent No. 6,199,059 to Dahan et al.
Ex. 1012	U.S. Patent No. 6,738,975 to Yee et al.
Ex. 1013	Scheduling Order for <i>Isix IP LLC v. SAP America, Inc.</i> , 2:23-cv-00237 (E.D.TX.)
Ex. 1014	U.S. Patent No. 6,151,608 to Abrams et al.
Ex. 1015	Curriculum Vitae of Dr. Henry Houh

I. INTRODUCTION

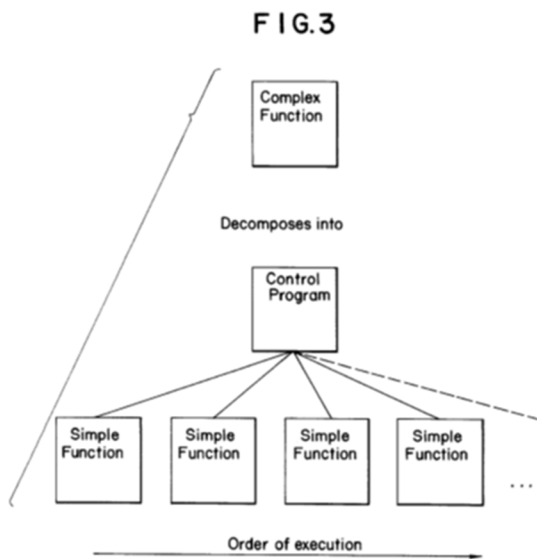
U.S. Patent No. 6,308,178 (the “178 patent,” Ex. 1001) is generally directed to integrating data among different software products, a concept long known and applied in business environments. The patent describes chaining together predefined subroutines that are stored in a repository in order to address “data synchronization (or updating) and migration” stemming from business organizations employing a variety of disparate computer applications, and to eliminate “the need for custom software . . . [and] manual data entry,” which can be “time consuming, [and] labor intensive.” *Id.* 1:14-37, 1:64-65, 3:13-20. The patent admits that “providing migration of data among heterogenous applications” was well-known, referring to two common categories of software that perform the same task: “Enterprise Application Integration” and “Enterprise Resource Planning.” *Id.* at 1:5-8, 1:38-43. The specification also identifies numerous products already in use that provide the same functionality, including offerings by SAP. *Id.* at 1:41-51.

The approach of chaining together data elements for integration was also known. U.S. Patent No. 5,842,205 to Brann (“Brann”) describes a system that works in the same manner to eliminate the need for custom code.



Ex. 1003, Fig. 4

Brann integrates data among source and target applications via a control program that activates subroutines (simple functions) in sequence to perform transformation of data from source to target.



Ex. 1003, Fig. 3

Also, U.S. Patent No. 6,208,990 to Suresh et al. (“Suresh”) describes integrating data among different software products. Suresh discloses a transformation engine that transforms data for integration between sources and targets. It does so using a library of transformation components that are arranged in sequence and then activated to perform the transformation.

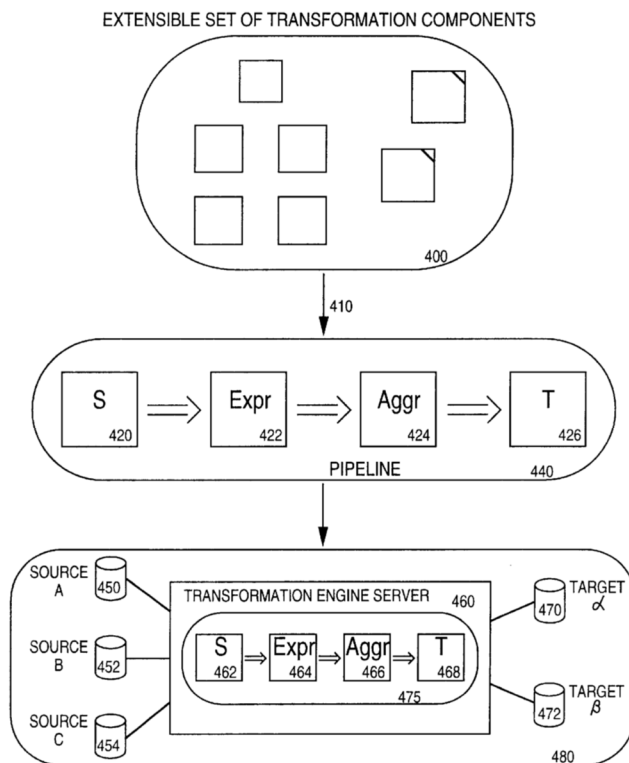


Figure 4

Ex. 1004, Fig. 4

Neither of these references was of record or considered during prosecution

of the '178 patent. Furthermore, the '178 patent received a first action allowance and there was no Office Action on the merits. Ex. 1005, pgs. 101-103.

There is nothing novel or non-obvious about the '178 patent's claimed system for obtaining data from a source, translating the data according to instructions, and storing the transformed data in a target. Accordingly, pursuant to 35 U.S.C. §§311, 314(a), and 37 C.F.R. §42.100, SAP America, Inc. ("Petitioner") respectfully requests that the Board review and cancel as unpatentable under 35 U.S.C. §103 claims 1-5, 7-10, 19, and 23-25 (hereinafter, the "Challenged Claims") of the '178 patent.

II. GROUNDS FOR STANDING

Petitioner certifies that the '178 patent is eligible for IPR and that Petitioner is not barred or estopped from requesting IPR challenging the patent claims. 37 C.F.R. §42.104(a).

III. RELIEF REQUESTED AND THE REASONS FOR THE REQUESTED RELIEF

Petitioner asks that the Board institute a trial for *inter partes* review and cancel the Challenged Claims in view of the analysis below.

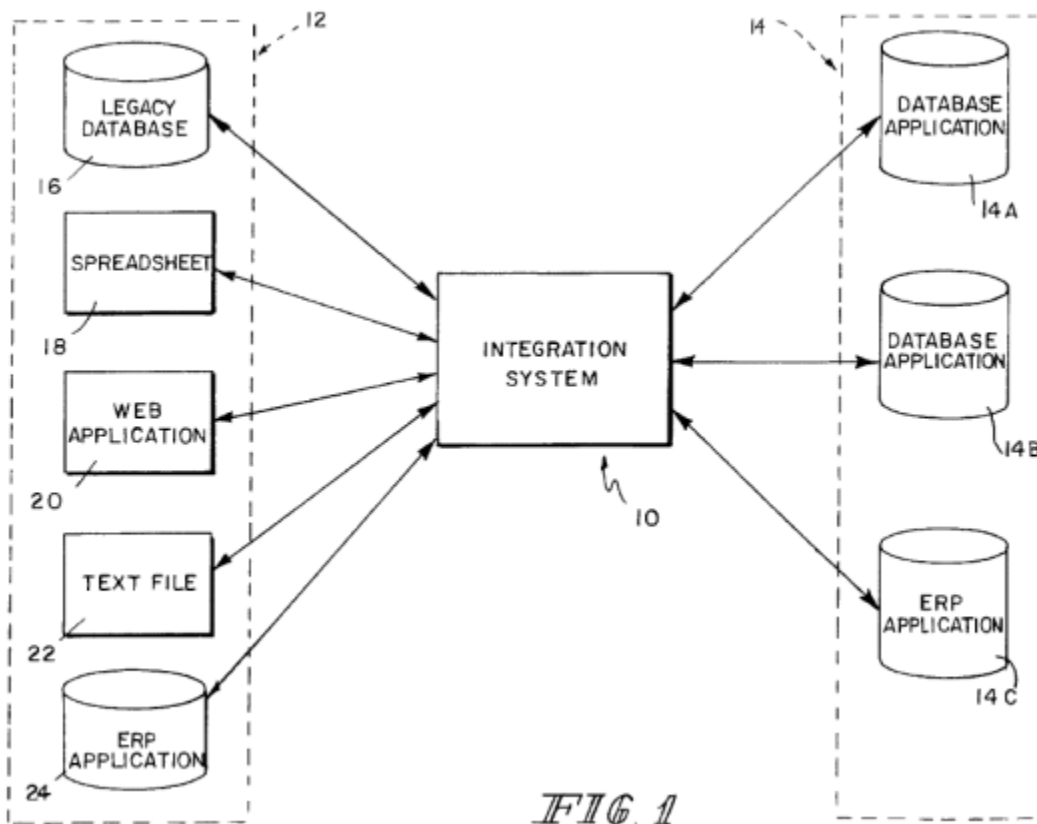
IV. SUMMARY OF THE '178 PATENT

As explained in the background, businesses often need "to integrate various business software applications employed by the diverse operating sectors of ... [the] organization (for example, financial operations, human resources, shipping

and receiving, etc.).” Ex. 1001, 1:14-25. Because the various sectors of a company may use different software applications, it has long been known to use systems for integrating the various software applications so that they interface properly. The ’178 patent claims such a system for integrating data among heterogeneous source applications and destination applications. Ex. 1001, 9:51-53.

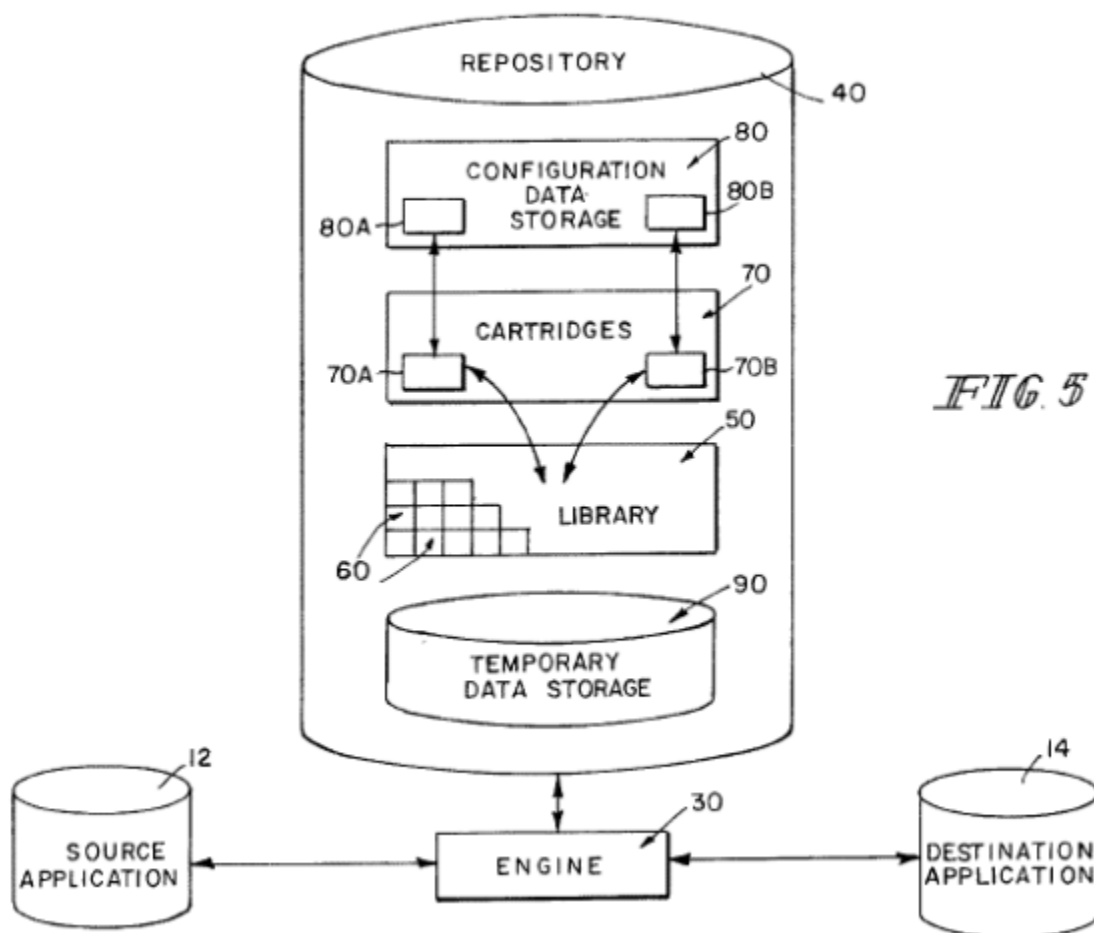
Independent claim 1 recites just two main elements—a repository and an active component. The “repository of information” is simply a store of information. Ex. 1001, 9:54-60. In particular, the repository includes “data elements” and “a first module.” The data elements are analogous to subroutines. *See Id.* at 6:33-37; Ex. 1002, ¶33. These subroutines provide different functions so that, rather than creating a custom program for integration, the repository 40 simply includes a library 50 of subroutines that may be combined as needed. A first module 70 in the repository 40 stores instructions sets for activating the data elements 60, that is, these instruction sets define which subroutines are called in what order. Ex. 1001, 6:54-60, 9:54-60, *Id.* The “first module” is a pre-built integration package for a specific application. Ex. 1001, 2:47-51. The first module includes rules for how the subroutines are combined so that the program just picks from the library which subroutines to use for data integration with the particular destination application, instead of providing a custom program that includes only the necessary subroutines. *Id.* at 2:53-56, 3:5-16.

Regarding the “active component,” that element includes an “input” and an “output.” Ex. 1001, 9:61-64. The input simply receives the data from the source application 12. *Id.* The output simply populates the destination application 14 with that data processed by the data elements 60 called from the repository 40. *Id.* at 2:56-60, 9:61-67.



Ex. 1001, Fig. 1.

A source application 12 is simply the application that has the data which needs to be integrated with another application, which is the destination application 14. Ex. 1001, 4:7-12.



Ex. 1001, Figure 5.

V. PROSECUTION HISTORY

The '178 patent was filed October 21, 1999, as U.S. Application No. 09/442,089. The '178 patent received a Notice of Allowance on August 14, 2001. Ex. 1005, pgs. 101-103. No substantive Office Action was issued.

In the Notice of Allowance, the Examiner stated:

Reasons For Allowance

4. The following is an examiner's statement of reasons for allowance:

The prior art of record does not teach or fairly suggest a system in combination elements as recited in claims 1, 23 and 34 wherein, an interfacier for populating the destination applications with data from the temporary data storage after the data is processed by the active component, the active component processing the data by communicating with the first module and activating data element chains according to instruction sets of the first module.

Ex. 1005, pg. 102.

Based on these reasons for allowance, the patent issued October 23, 2001.

Notably, the limitations referenced by the Examiner, i.e., “an interfacier for populating the destination applications with data from the temporary data storage after the data is processed by the active component, the active component processing the data by communicating with the first module and activating the data element chains according to instruction sets of the first module” **are not recited** in independent claim 1.

In other words, independent claim 1 (and dependent claims 2-22) *do not include* the elements considered by the Examiner to distinguish over the prior art.

VI. LEVEL OF ORDINARY SKILL IN THE ART

A person of ordinary skill in the art (“POSITA”) in the field of the ’178 patent, as of its earliest possible priority date of October 21, 1999, would have a

bachelor's degree in computer science, computer engineering, or equivalent training, and approximately two years of experience working in the computer science or engineering field, or with integration systems. Lack of work experience can be remedied by additional education, and vice versa. Ex. 1002, ¶¶17-20.

VII. CLAIM CONSTRUCTION

Various claim terms were construed in a prior litigation asserting the '178 patent. Ex. 1007. Patent Owner relied on that claim construction in a prior IPR for the '178 patent, IPR2022-01488.¹ Ex. 1008, 20. Petitioner relies on Patent Owner's position applying the district court's claim construction for the purposes of this proceeding. No special construction is required for other claim terms.²

In the prior district court case, the term “data element(s)” was construed as “individual unit(s) of reusable code.” Ex. 1007, 2. The preamble was determined to be limiting. *Id.* Various claim elements were determined not to be interpreted as means-plus-function elements. *Id.* at 2-3. All other terms proposed for construction were given their plain and ordinary meaning. *Id.*

VIII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

¹ IPR2022-01488 was filed by a different Petitioner and settled prior to any institution decision.

² Petitioner accepts Patent Owner's constructions for purposes of this proceeding only to simplify issues. Petitioner reserves the right to propose alternative constructions in the corresponding District Court case.

A. Challenged Claims

Petitioner challenges claims 1-5, 7-10, 19, and 23-25.

B. Statutory Grounds for Challenges

Grounds	Claims	Basis
#1	1, 3, 4, 7, 8	§103 over Brann
#2	1 – 5, 7, 9, 10, 23 – 25	§103 over Suresh
#3	1 – 5, 7, 9, 10, 23 – 25	§103 over Suresh in view of Brann
#4	8 and 19	§103 over Suresh in view Beauchamp
#5	8 and 19	§103 over Suresh in view of Brann and Beauchamp

U.S. Patent No. 5,842,205 to Brann (“Brann”, Ex. 1003) was filed December 2, 1993, and issued November 24, 1998. Brann is prior art under pre-AIA 35 U.S.C. §102(e).

U.S. Patent No. 6,208,990 to Suresh et al. (“Suresh”, Ex. 1004) was filed July 15, 1998, and issued March 27, 2001. Suresh is prior art under pre-AIA 35 U.S.C. §102(e).

U.S. Patent No. 6,621,505 to Beauchamp et al. (“Beauchamp”, Ex. 1006) was filed September 30, 1998, and issued September 16, 2003. Beauchamp is prior art under pre-AIA 35 U.S.C. §102(e).

C. Prior Art Overview

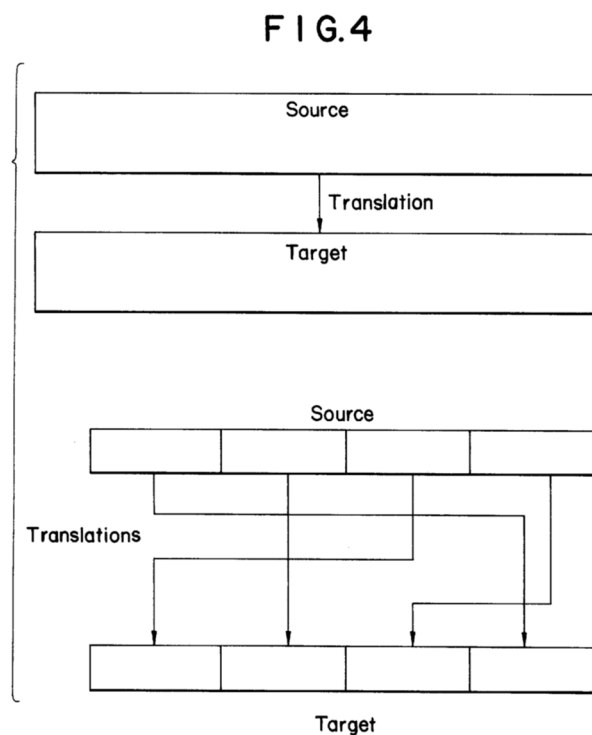
1. Summary of Brann

Brann, assigned to a subsidiary of Petitioner SAP America, Inc., was not of record or applied during prosecution of the '178 patent. Brann teaches a system for integrating data among source and destination applications. Ex. 1003, 1:10-13, 5:27-28; Ex. 1002, ¶46. A control program activates subroutines, stored in a database, in a specified order to transform data from the source to the target. Ex. 1003, Abstract, 7:27-30; 7:55-57. The control program accesses the database to obtain its instructions. *Id.* The database stores information about the source and target, as well as all information needed for the transformation. Ex. 1001, Abstract, 3:49-50; Ex. 1002, ¶46.

More specifically, Brann describes a mechanism for “the rapid integration of computer systems.” Ex. 1003, 5:27-28. Just as in the '178 patent, Brann recognized the problem that “[o]ver time, the number of independent systems has grown enormously. There has been little attention paid to coordination of the data between the competing programs.” *Id.* at 1:16-30; Ex. 1002, ¶47. Brann also recognized, just like the '178 patent, that using a new computer program as a go-between for heterogeneous applications requires custom, manual programming. *Id.* at 2:59-62; Ex. 1002, ¶47. (*See* '178 patent, Ex. 1001, 1:55 – 2:23).

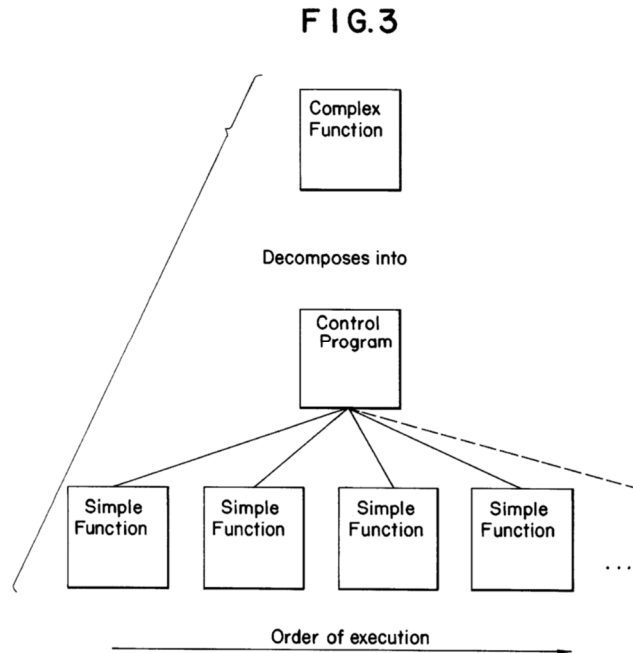
The system of Brann solved the problem of integrating data between a heterogeneous source application and destination application prior to and in the

same manner as the '178 patent. Ex. 1002, ¶46-50. Figure 4 of Brann reproduced below illustrates Brann's integration of data among source and target applications.



Ex. 1001, Fig. 4.

Brann utilizes the control program to call subroutines that perform simple functions to carry out the transformation of data between source and target.



Id. at Fig. 3.

Brann states that the simple functions for transformations “are best implemented as subroutines which can be re-used in similar instruction needs.” *Id.* at 6:29-36. The “subroutines are capable of being assembled into any conceivable Translation Program.” *Id.* at 9:13-14.

Brann also describes a repository in the form of a database. *Id.* at Abstract, 10:58-11:30. The database stores all the “data and instructions corresponding to the steps necessary to translate the source message to the target message.” *Id.* at 4:6-8; Ex. 1002, ¶50.

2. Summary of Suresh

Suresh was not of record or applied during prosecution of the '178 patent. Just like the '178 patent, Suresh is directed to integrating data among applications through a “data extraction/transformation/loading (ETL) process.” Ex. 1004, 4:55-56.

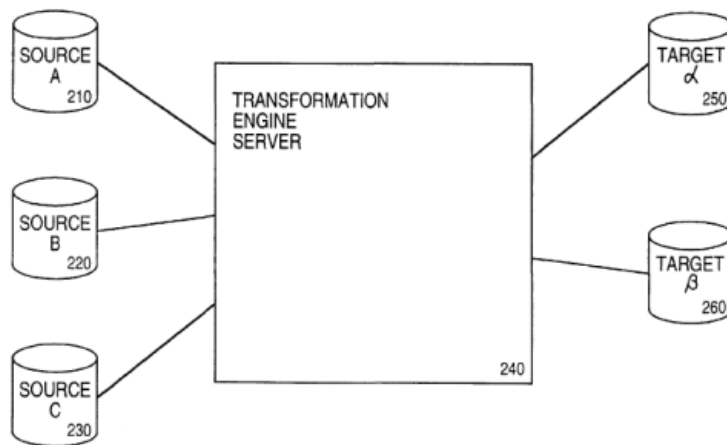


Figure 2

Id. at Fig. 2.

The Source A, Source B, and Source C are “operational databases 210, 220, and 230 [that] store the data resulting from business and financial transaction, and/or from equipment performance logs.” *Id.* at 6:8-11. The operational databases 210, 220, and 230 “provide the data source from which the transformation engine server 240 can extract, transform, then load the data into the databases 250 and 260.” *Id.* at 6:15-20. The “[d]atabases 250 and 260” are “warehouses or data marts” and are shown as Target α and Target β in Figure 2.

Id. at 6:16-19.

The transformation engine server performs the ETL process using a “cascading sequence of transformation components,” called a pipeline. As illustrated in Figures 4-9, there may be multiple pipelines that include different cascading sequences of transformation components. Ex. 1004, Figures 4-9. The transformation components are “ready-made” pieces of source code where each transformation component “is responsible for one specific type of transformation.”

Id. at 6:46-52; 6:61-65.

Suresh’s system also includes a repository. Ex. 1004, 6:23. The repository keeps “track of session information” and stores all “mapping information relating to how data is to be mapped and transformed from target tables of the operational databases to target tables of data marts.” *Id.* at 6:23-29.

D. Ground 1: Claims 1, 3, 4, 7 and 8 are obvious under 35 U.S.C. §103 over Brann.

1. Claim 1

[1.0] *A system for integrating data among heterogeneous source applications and destination applications, comprising:*

If the preamble is limiting, Brann describes and suggests the claimed “*system for integrating data*” as it describes a “process for providing an efficient system for assembling and for translating information from one computer to another.” Ex. 1003, 1:10-13, 5:27-28 (“competing programs”). The system

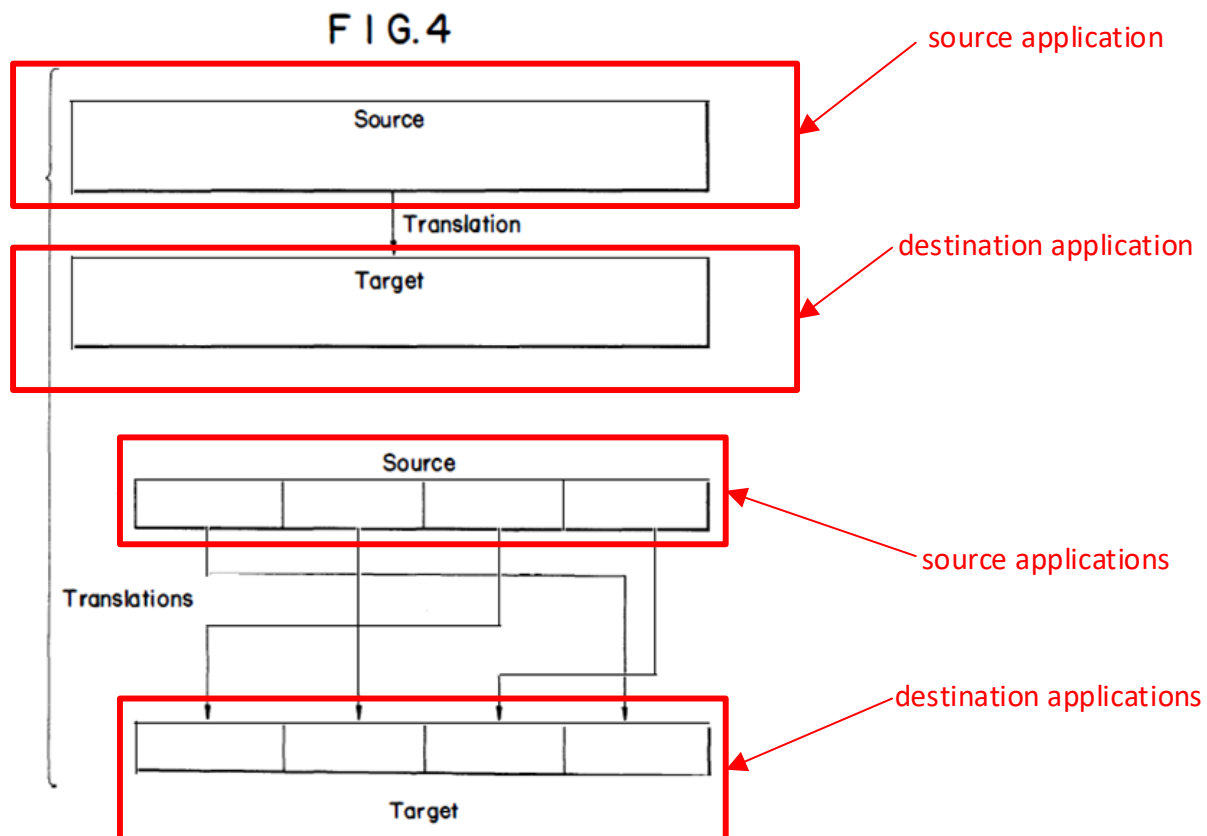
includes “a control program” and “a database.” *Id.* at 21:47-58. The control program executes on a computer and accesses the database to look up and obtain information, thus forming a system. *Id.*; *see also* Ex. 1003, 13:33-34; Ex. 1002, ¶56.

Brann describes and suggests integrating “*data among heterogeneous source applications and destination applications.*” Ex. 1002, ¶57-62. Brann describes and suggests integrating data among applications by addressing the “need for a mechanism which will aid in the rapid integration of computer systems” and describing a “computer programming method ... for translating source messages to target messages” and “coordination of the data between the competing programs.” Ex. 1003, 5:27-28, Abstract, 1:16-30. A POSITA would recognize the “competing programs” described in Brann are “*applications*,” as “program” and “application” were used interchangeably at the relevant time. Ex. 1002, ¶58. *See also*, Ex. 1009.

In addition, Brann describes and suggests that the described computer systems and applications are of “different types,” that is they are heterogeneous. Ex. 1002, ¶59. For example, Brann states that the programs are executed on “different” and “independent” computer systems. Ex. 1003, 2:54-55, 1:16-30. A POSITA would recognize that since the computer systems are heterogeneous, the applications running on those heterogeneous computer systems are also heterogeneous. Ex. 1003, 2:45-55. Ex. 1002, ¶59-60. For example, data is

transformed when an upgrade is performed or moved to a different, heterogeneous platform. Ex. 1002, ¶¶59-60

Moreover, Brann discloses that data sent from source applications to target applications is translated or transformed. Ex. 1003, 11:7-30. There would be no need to translate and/or transform the data if the source and target were the same. A POSITA would recognize that the source and target of Brann are heterogeneous. Ex. 1002, ¶¶ 59-62. Annotated Figure 4 of Brann below illustrates integrating data among source applications and destination applications via translation(s).



Ex. 1003, Fig. 4 (annotated); *See also* Ex. 1003, 7:29-52; 11:46-47; 14:33-34.

As such, a POSITA would understand Brann also describes or suggests that these heterogeneous applications are “*heterogeneous source applications and destination applications.*” Ex. 1002, ¶¶56-62.

[1.1] *a repository of information relating to the source applications and the destination applications, the repository including a plurality of data elements each being capable of performing a discrete operation on a piece of data, and*

Brann renders this claim element obvious as it describes and suggests a database storing information related to source and target applications, and storing subroutines used to transform data among those applications. Ex. 1002, ¶¶63-78.

Just like the ’178 patent, Brann discloses a database as “*a repository of information.*” Ex. 1003, 4:6-8. The ’178 patent equates a repository with a database: “[c]onfiguration data of configuration data storage 80, data elements 60 of library 50, and temporary data of temporary data storage 90 are all stored as Oracle tables within a relational **database represented by repository 40.**” (emphasis added). Ex. 1001, 4:33-37. A POSTIA would recognize this understanding. Ex. 1002, ¶¶63.

In a similar manner, the database in Brann stores “*information relating to the source applications and the destination applications.*” For example, Brann describes “storing instructions, data, and the structures of any source stimuli to be acted upon in the database.” Ex. 1003, 3:49-51. A POSITA would understand this to be information relating to the source and destination applications. Ex. 1002,

¶¶64.

In addition, in Brann, information describing the message types used by the source and target, which is information relating to the source and destination applications, is stored in tables in the database:

Each of the incoming message types is recorded in the source table together with any arbitrary data to be found in messages of this type in the section defined in the location table. Each of the target messages for each source message is recorded in the target table. Each of the details of the transformation is recorded in a translation details table.

Ex. 1003, 11:4-10.

The database also stores “data and instructions corresponding to the steps necessary to translate the source message to the target message.” *Id.* at 4:6-8. The above types of information stored in the database are also “*information relating to the source applications and the destination applications.*” Ex. 1002, ¶¶65-66.

The database of Brann also “*include[s] the plurality of data elements each being capable of performing a discrete operation on a piece of data.*” The claim term “data element(s)” was construed as “individual unit(s) of reusable code.” Ex. 1007, 2. Just like the ’178 patent, Brann describes and/or suggests reusable subroutines (*data elements*) that perform discrete operations. Ex. 1002, ¶¶67-68. For example, the ’178 patent states that the data elements “essentially consist of

reusable code or units of work that perform a discrete data manipulation function, and may be shared” Ex. 1001, 6:33-35. In the same manner, Brann teaches “an atomic unit of processing appropriate to the task in hand. In fact, in the construction of computer systems, the form of the ‘instructions’ is usually defined by the task at hand and is then expressed using a suitable programming language. It is well known that such functions are best implemented as subroutines which can be re-used in similar instruction needs.” Ex. 1003, 6:29-36. Brann thus describes and/or suggests the claimed “*data elements*.” Ex. 1002, ¶67-68.

The subroutines of Brann are “*each being capable of performing a discrete operation on a piece of data*.” Brann states that a subroutine “carries out the particular transformation required.” *Id.* at 11:11-12. Brann states that respective subroutines may be used “[t]o carry out functions like:

Copy (with justification to left or right, as appropriate)

Convert from character to integer numeric formats (and vice versa)

Convert from character to floating point numeric formats (and vice versa)

Convert from floating point to integer numeric formats (and vice versa)

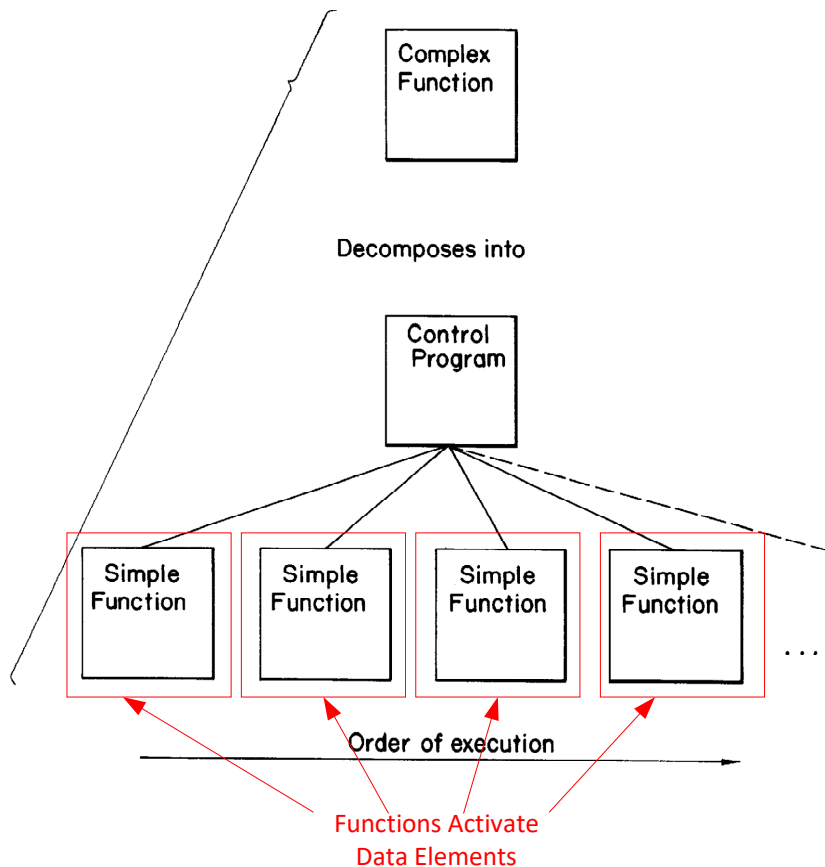
Lookup incoming data file in table of values (output result as characters, integer, or floating point)

A further list of representative functions is attached in the appendix. When invoked, each of these routines is passed a reference to the data it must take as

input, a reference to the place its output must be written to, the maximum allowable lengths of input and output, and a reference to the lookup table (if any) being employed.” *Id.* at 8:38-48.

“Translation Methods are written as subroutines which receive a particular set of parameters. Each Method carries out a particular type of transformation. For example, there is one to do copying and one to do conversion of character format numbers into integers. Each of the Methods carries out a particular type of manipulation of simple data types.” *Id.* at 12:47-52.

FIG. 3



Ex. 1003, Fig. 3 (annotated).

Brann illustrates the simple functions (subroutines) invoked by the control program to perform an operation on data. Brann thus describes and/or suggests that the subroutines are used to perform “*discrete operations on pieces of data.*” Ex. 1002, ¶¶70-73.

Moreover, a POSITA would understand the “task in hand” and “particular transformation” performed by the subroutines in Brann are types of “*discrete operations.*” A subroutine is a discrete operation as it is labeled as such. This is the

case regardless of what the subroutine is composed of, for example, various sub-operations. Ex. 1002, ¶¶74 Brann thus renders obvious “*data elements*” (subroutines) that perform a “*discrete operation on a piece of data*” (task in hand, etc.). Ex. 1002, ¶¶70-74.

Brann also describes and/or suggests that the subroutines are included in the database. Ex. 1002, ¶¶76-78. As described above, Brann describes and suggests that the subroutines, also called basic programming elements, are used to carry out the transformations. Ex. 1003, 7:22-30. Brann states that these subroutines are stored: “It is still a further object of this invention to provide a method of creating programs by storing and calling upon basic programming elements and combining those elements for use at the program's execution-time.” *Id.* at 3:5-8. Brann also states the steps for the transformation are described in the “database in which all necessary instructions and data are stored.” *Id.* at Abstract. Brann describes throughout that all of the instructions and data needed to perform a translation are stored in the database. *See* Ex. 1003, Abstract, 3:5-8, 3:19; 4:6-7, 4:53-55, 14:63-64, 20:64-67. As such, a POSTIA would understand that Brann describes and/or suggests that the subroutines are stored in the database. Ex. 1002, ¶¶77-78. Brann thus renders this claim element obvious. Ex. 1002, ¶¶63-78.

[1.2] *[the repository including...] a first module including a plurality of instruction sets, each instruction of each instruction set activating a data element; and*

Brann renders this claim element obvious as the database (*repository*) of Brann includes a plurality of transformation instructions (*instructions sets*) for the control program to activate subroutine(s) (*data element(s)*) to perform the translation. Ex. 1002, ¶¶79-89. The transformation instructions are stored in the database, for example in rows of a database table, which a POSITA would understand corresponds to the “*first module*.” Ex. 1002, ¶¶79-83.

The “*first module*” is a collection of instructions sets. Ex. 1001, 2:53-57, 8:55-56. The instruction sets of Brann include a “list of subroutines to be invoked and an order in which they should be invoked. The Control Program then invokes the required subroutines in the required order,” Ex. 1003, 7:26- 27. These instructions are stored in the database. *Id. at* 7:33-34. This “list of subroutines to be invoked and an order in which they should be invoked” stored in the database teaches and/or suggests the “*first module including a plurality of instruction sets*.” Ex. 1002, ¶¶79-83.

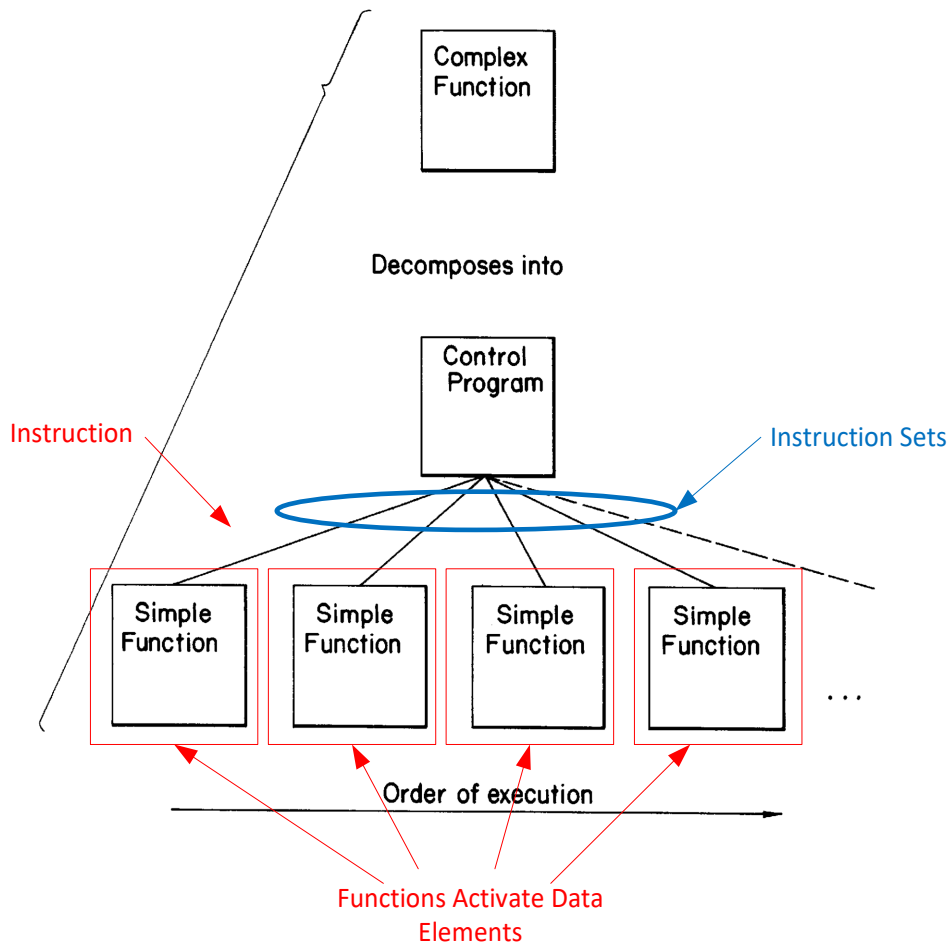
Brann also teaches and/or suggests “*each instruction of each instruction set activating a data element*” as each of the instructions activates a subroutine by instructing the Control Program to call that subroutine. Ex. 1002, ¶¶84-87. The ’178 patent describes this function as “[m]odule 70A provides an instruction set for achieving the output which then calls upon a series of data elements 60 within library 50 to perform individual tasks that may be common to a variety of modules

70.” Ex. 1001, 6:54-58. “[M]odules 70 do not actually contain data elements 60, but rather contain information describing where they may be found in library 50.” *Id.* at 7:38-41.

In a similar manner, Brann discloses “the database contains a unique string of characters for each available subroutine.” Ex. 1003, 7:45-46. “That name is used by the Control Program to derive the executable function reference of the appropriate subroutine when it reads the database.” *Id.* at 11:13-17. The control program “resolves the reference to find the subroutine and execute [sic] the subroutine.” *Id.* at 11:19-20. Thus, the database and database tables contain information describing where the subroutines to be activated may be found, just as the “module” described in the ’178 patent. Ex. 1002, ¶¶84-85.

Annotated Fig. 3 of Brann below illustrates how the instructions in the instruction set are used to activate a data element. The line between the control program and each simple function represents that the control program calls a sequence of these simple functions, which are “a sequence of instructions” that execute in a particular order. Ex. 1003, 6:25-36, 5:15-16, Fig. 3. A POSITA would recognize that the transformation program, database tables and rows within those tables of Brann as the claimed “*first module including a plurality of instruction sets ... activating a data element(s).*” Ex. 1002, ¶¶86-88.

FIG. 3



Ex. 1003, Fig. 3 (annotated).

Thus, the database of Brann storing all data and instruction necessary for the translation, including the “list of subroutines to be invoked and an order in which they should be invoked,” and “the reference to find the subroutine and execute the subroutine” renders obvious “[the repository including...] a first module including a plurality of instruction sets, each instruction of each instruction set activating a data element.” Ex. 1002, ¶¶79-89.

[1.3] *an active component connected to the repository having an input for loading data from the source applications, and an output for populating the destination applications with processed data, the active component processing the data from the source applications by activating data elements according to the plurality of instruction sets in the module.*

Brann renders this claim element obvious as it describes a control program that communicates with the database. The control program receives data from the source application via an input, transforms the data by activating subroutines according to the instructions in the database, and provides the processed data to the target application via an output. Ex. 1002, ¶¶90-99.

Brann teaches and/or suggests the “*active component connected to the repository*” as it describes a “control program, executing on a computer, communicating with the database.” Ex. 1003, 7:33-34; Ex. 1002, ¶¶90-91. “The Control Program is a program which invokes basic computing steps in the form of Methods. It calls upon databases to tell it or instruct it as to which of these Methods should be used under which circumstances of incoming data.” Ex. 1003, 14:20-23. “The Control Program extracts its instructions from the database.” *Id.* at 7:33-35 Thus, Brann describes and/or suggests the “*active component connected to the repository.*” Ex. 1002, ¶¶90-91.

To the extent Brann does not include a figure depicting the database connected to the control program, POSITAs would have known that the database and control program would be connected. Ex. 1002, ¶92, citing Ex. 1010.

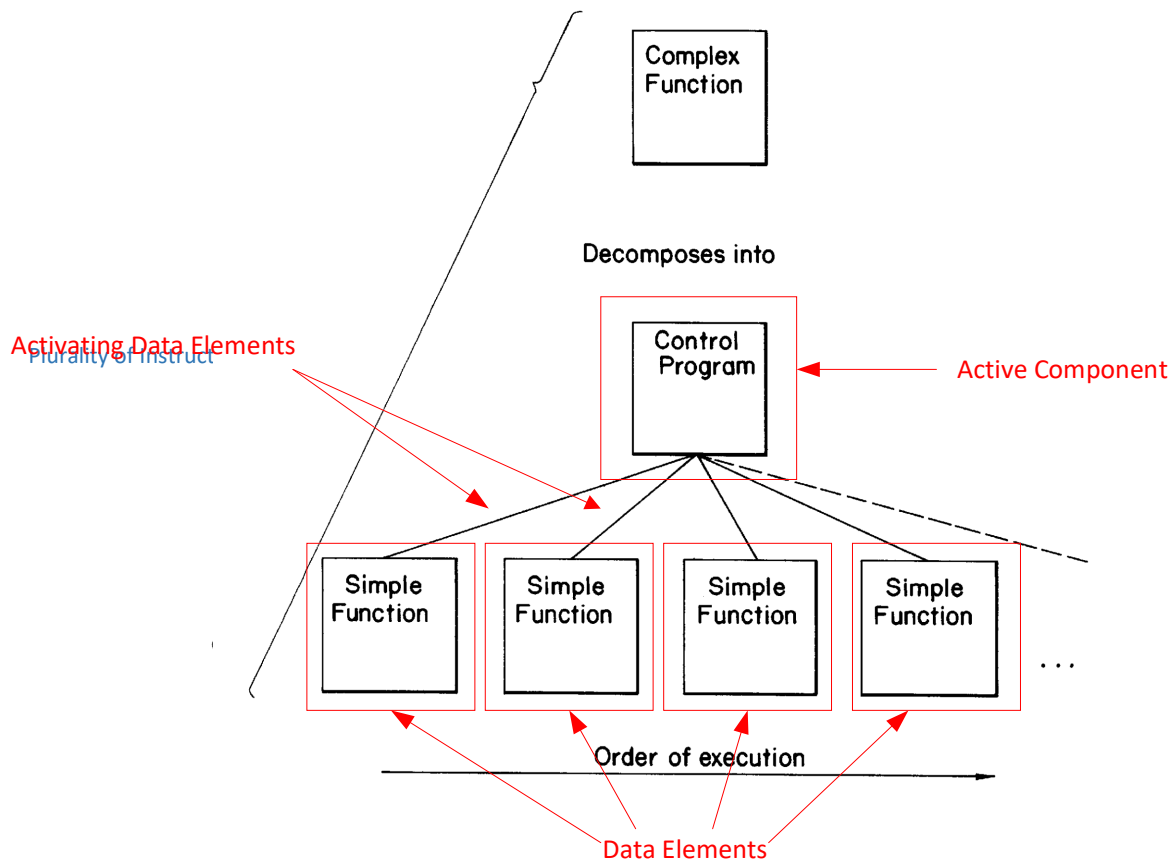
Brann teaches and/or suggest the “*active component ... having an input for loading data from the source applications*” because it describes “providing the source messages to the control program.” *Id.* at 3:16; Ex. 1002, ¶93. “The Control Program exists in a world where it is receiving Stimuli. In effect it is being given incoming data to operate upon.” *Id.* at 11:31-33. “[T]he Control Program receives an incoming message, a stimulus.” *Id.* at 12:41-42. A POSITA would recognize that the control program would have an input to receive the source messages or Stimuli. Dec. cite. A POSTIA would have known the “source” messages or Stimuli include data from the “source” application, as indicated by the name “source” message. Ex. 1002, ¶93. In addition, Brann discloses a “method of efficiently translating source messages to target messages” Ex. 1003, 2:65-68. “Once the data to be translated has been obtained (**i.e., source data**), it must be identified.” *Id.* at 8:6-7 (emphasis added). As such a POSTIA would have understood that “source messages” included “source data” from the source application, and that Brann teaches and/or suggests “*the active component has an input for loading data from the source application.*” Ex. 1002, ¶93.

Brann also teaches and/or suggests the active component having “*an output for populating the destination applications with processed data*” as Brann describes that the transformed message (*processed data*) is provided to the target (*destination application*). Ex. 1002, ¶94. “When the Control Program has executed

all the Methods it is directed to do, it delivers the outgoing structure to the receiver.” Ex. 1003, 12:6-10. “The control program, responding to the database and the translation method programs, then delivers the completely translated target message to the receiver.” *Id.* at Abstract. A POSITA would thus recognize that Brann describes and/or suggests that the control program (*active component*) has an output for populating destination applications with processed data. Ex. 1002, ¶94.

Brann also describes and suggests that the control program processes the data from the source message based on the instructions in the database. The instructions in the database set forth which subroutines (or methods) are to be called, and the order in which to call them, to perform the required transformation. Ex. 1002, ¶95-98. “The Control Program extracts its instructions from the database.” Ex. 1003, 7:33-34. “The Control Program is a program which invokes basic computing steps in the form of Methods. It calls upon databases to tell it or instruct it as to which of these Methods should be used under which circumstances of incoming data.” *Id.* at 14:20-23.

FIG. 3



Ex. 1003, Fig. 3 (annotated).

“The control program, responding to the database and the translation method programs, then delivers the completely translated target message to the receiver.”

Id. at Abstract.

Thus, that the control program of Brann (i) is connected to the database, (ii) has an input for loading data from the source applications, (iii) has an output for populating data in the target applications, and (iv) transforms the source’s data by activating subroutines according to the instructions stored in the database renders obvious “*an active component connected to the repository having an input for*

loading data from the source applications, and an output for populating the destination applications with processed data, the active component processing the data from the source applications by activating data elements according to the plurality of instruction sets in the module.” Ex. 1002, ¶¶90-99.

2. Claim 3

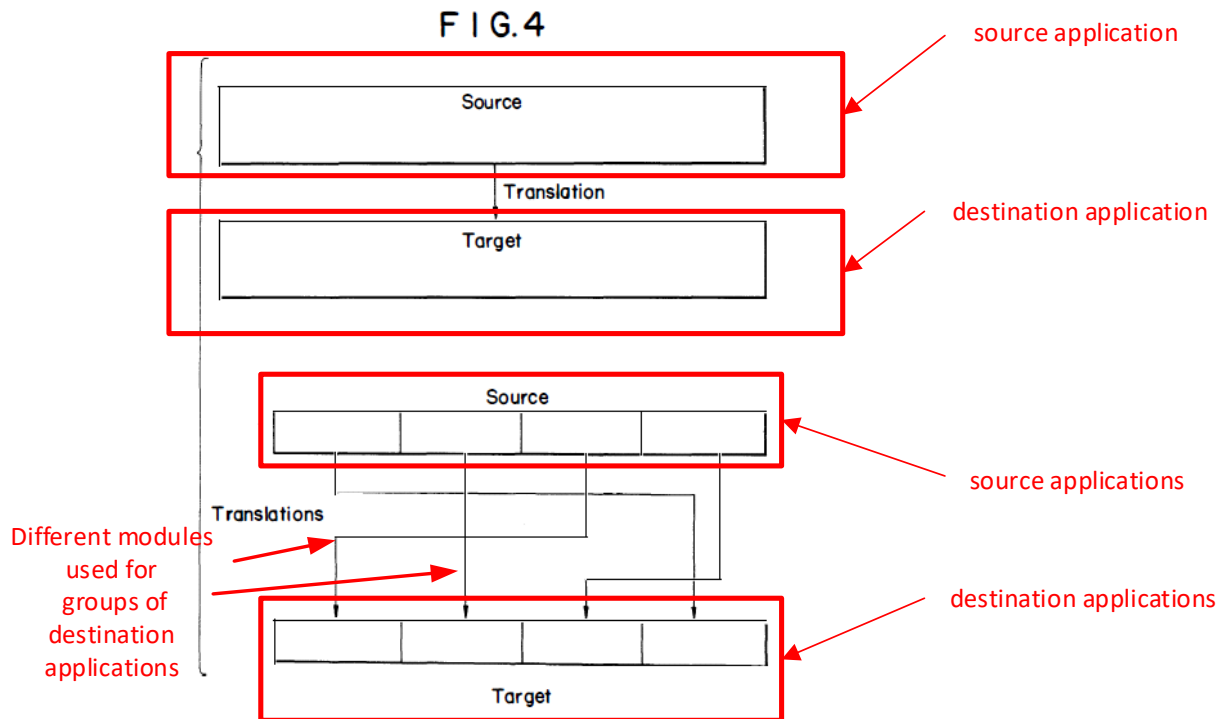
[3.1] *The system of claim 1 further comprising a second module including a plurality of instruction sets, each instruction of each instruction set activating a data element,*

As discussed in [1.2], Brann describes and/or suggests a database storing a *first module* as “a set of transformation instructions.” The set of transformation instructions forms a translation program. Brann describes that a “multiplicity of translation programs” may be provided. Ex. 1003, 15:25-27. Thus, Brann suggests this claim element. Ex. 1002, ¶¶100-102.

[3.2] *the first module plurality of instruction sets being defined for interface with a first group of destination applications, the second module plurality of instruction sets being defined for interface with a second group of destination applications.*

Brann describes and suggests that each transformation (*first module and second module*) can provide data to different targets (*a first group of destination applications and a second group of destination applications*) because Brann’s

Figure 4, for example, illustrates that different transformations (*first module*) interface with one target (*first group of destination applications*) and another transformation (*second module*) interfaces with another target (*second group of destination applications*):



Ex. 1003, Fig. 4 (annotated); Ex. 1002, ¶103.

Brann describes that a multiplicity of translation programs may be provided. Ex. 1003, 15:25-28. Because Brann discloses that data from a transformation can be stored in one or more targets (Ex. 1002, ¶104) and that different types of transformations may be performed by a multiplicity of translation programs (Ex. 1003, 11:4-19, 59-60), a POSITA would have found it obvious to implement

different transformation instructions that each interface with different groups of one or more targets. Thus, Brann renders obvious the recited limitation. Ex. 1002, ¶¶100-104.

3. Claim 4

[4] *The system of claim 3 wherein at least one data element of the plurality of data elements is activated by both the first module plurality of instruction sets and the second module plurality of instruction sets.*

First, as discussed in [1.1], Brann discloses a set of ready-made subroutines (*data elements*). Ex. 1003, 6:56-65. Transformations can be carried out using the set of subroutines. For example, “Translation Methods are written as subroutines which receive a particular set of parameters. Each Method carries out a particular type of transformation. For example, there is one to do copying and one to do conversion of character format numbers into integers. Each of the Methods carries out a particular type of manipulation of simple data types.” *Id.* at 12:47-52.

Brann further describes and/or suggests that different translation programs can each activate a common subroutine, particularly since the subroutines are being selected from a set of subroutines. Ex. 1003, 7:21-30, 8:36-50. A subroutine is invoked using “a reference to the subroutine code (usually the function name or a function pointer).” *Id.* at 8:61-65. It is well-known that subroutines “can be re-used in similar instruction needs.” *Id.* at 6:34-37.

In addition, the selected subroutine would be “*activated by both the first*

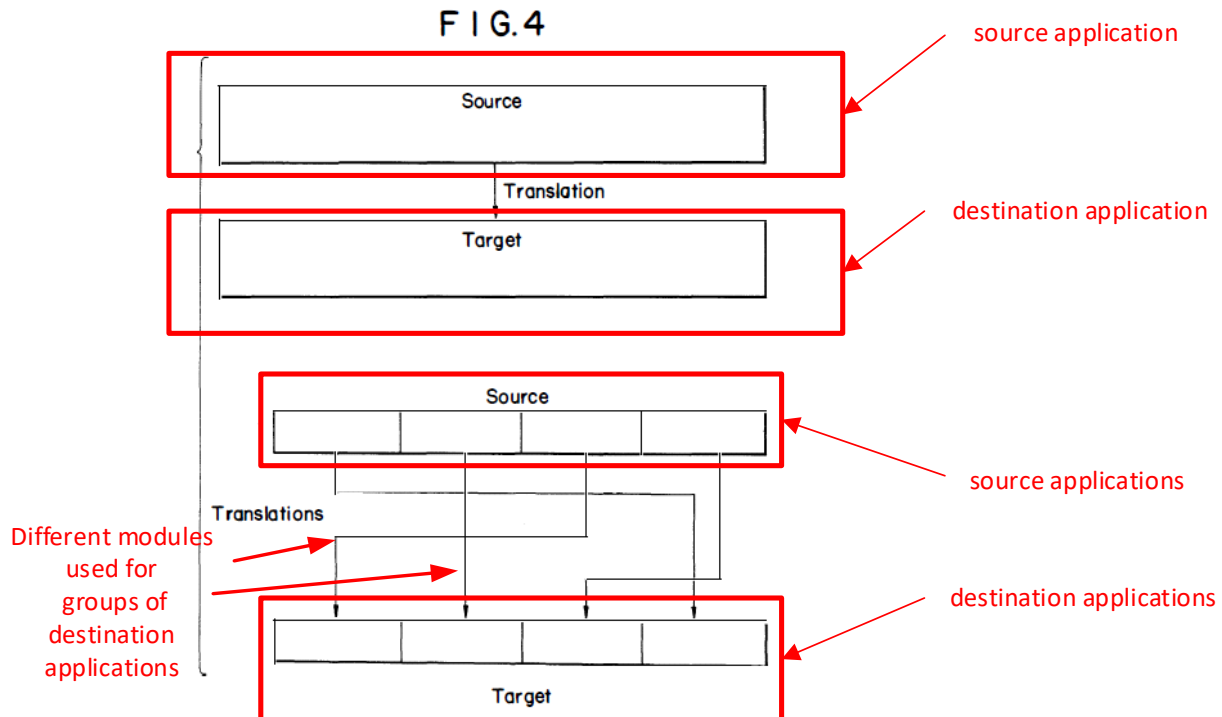
module plurality of instruction sets and the second module plurality of instruction sets” as discussed in [1.3] and [3.1]. Specifically, Brann discloses that the subroutine would be *activated* during the transformation process. Ex. 1003, 11:10-20.

Thus, the at least one subroutine of Brann that (i) is part of a set of subroutines, (ii) can be re-used in different translation programs, and (iii) is activated during the transformations renders obvious the recited limitation. Ex. 1002, ¶¶105-109.

4. Claim 7

[7] *The system of claim 1 wherein the repository is capable of receiving a plurality of modules.*

Brann describes and/or suggests a repository of information, i.e. database, used to store a plurality translation programs. See [1.2]. For example, Brann discloses “storing instructions, data, and the structures of any source stimuli to be acted upon in the database.” Ex. 1003, 3:49-51. Brann illustrates that different translations may be used for integrating data between different sources and targets.



Brann describes that a multiplicity of translation programs may be provided. Ex. 1003, 15:25-28. A POSTIA would understand that the database (*repository*) of Brann is capable of receiving a multiplicity of translation programs (*plurality of modules*), rendering this claim element obvious. Ex. 1002, ¶¶110-112.

5. Claim 8

[8] *The system of claim 1 wherein the destination applications are included within an ERP product.*

As admitted in the '178 patent, ERP products were well known to a POSITA at the relevant time. Ex. 1001, 1:38-47. ERP products provide applications for business functions such as accounting, purchasing, and other financial matters. *Id.* Brann discusses an example of an integration problem and financial data that are

typical of ERP systems. ("An example of this is the conversion of a monetary currency code from one arbitrary form (e.g., that defined by Reuters) into another (e.g., that defined by S.W.I.F.T.). Ex. 1003, 8:25-28. This type of financial application would typically be part of an ERP product. Ex. 1001, 1:38-47

Accordingly, a POSITA would understand the transformations from a monetary currency code from one arbitrary form into another for use by the target application as described in Brann to render obvious the recited limitation. Ex. 1002, ¶¶113-116.

E. Grounds 2 and 3: Claims 1 – 5, 7, 9, 10, 23 – 25 are obvious under 35 U.S.C. §103 over Suresh alone or combined with Brann.

Grounds 2 and 3 are discussed together below. Brann is used to supplement Suresh for elements 1.0 and 23.0 (applications) and 1.2 and 23.4 (first module).

1. Reasons to Combine Suresh with Brann

A POSITA would have considered the teachings of Brann and Suresh because they are analogous prior art in the same field of endeavor as the '178 patent – integrating data among multiple, disparate sources and targets. Ex. 1002, ¶¶117; Ex. 1003, 1:10-13, 5:27-28; Ex. 1004, 4:55-57, 6:15-19.

a.) Integrating data between applications

To the extent Suresh's Sources and Targets are not applications, it would have been obvious at the time of the alleged invention to provide applications instead of databases or data warehouses in view of Brann. Suresh would have

benefited from Brann’s teachings. Any front-end applications for the databases in Suresh may be integrated via the system of Brann. Ex. 1002, ¶¶118. For example, Brann describes the problem of communication between different types of computer programs, and provides a solution to that problem. Brann provides “an efficient system for assembling and for translating information from one computer to another.” Ex. 1003, 1:10-13. A POSITA would have understood that the databases or data warehouses of Suresh could be replaced with applications to solve the problem of interoperability between different types of applications, as well as different databases. Ex. 1002, ¶118.

There would have been a reasonable expectation of success in the combination because a system for integrating data between heterogeneous applications was already known. Ex. 1003, 1:10-13; Ex. 1002, ¶¶118-119.

b.) *Storing instructions in a database*

To the extent Suresh does not suggest the first module including a plurality of instruction sets, Suresh would have benefited from Brann’s teachings so that the pipeline of Suresh could be quickly and easily reused or even modified for different transformations. Furthermore, adding the pipeline to the repository allows the pipeline to be more easily bound to one or more processes, and be more readily used to construct further more complex pipelines. For example, Suresh teaches that “a repository ... is used to keep track of ... mapping information relating to

how data is to be mapped and transformed from target tables of the operational databases to target tables of data marts.” Suresh, 6:25-29.. Ex. 1002, ¶120.

It would have been obvious to provide a first module including a plurality of instruction sets, each instruction of each instruction set activating a data element in Suresh, as it would be use of a known technique (storing instructions) to improve known devices or methods in the same way (allowing the instructions to be modified and/or saved) to yield predictable results (calling subroutines as set forth in computer instructions, with a reasonable expectation of success. Ex. 1002, ¶120. For example, it would have been obvious to do so to store and combine transformation components at run time, as taught by Brann. Ex. 1003, 3:9-15.

In addition, it would have been obvious to do so to eliminate the need for new programs and for manually creating the pipeline, as taught by Brann. Ex. 1003, 2:53-62. Once the pipeline is created and saved, updates may be made by revising the stored pipeline and by adding new instructions to pipelines in the database to create different transformations, as opposed to needing to create a new pipeline from scratch for every use or modification. Ex. 1002, ¶122. This also allows different computers to use the same, stored transformation. *Id.*

Accordingly, there would have been a reasonable expectation of success in the combination because techniques for storing transformation so that they may be quickly and easily reused or even modified techniques were already well-known

and applied in data integration. Ex. 1002, ¶¶120-123.

2. Claim 1

[1.0] A system for integrating data among heterogeneous source applications and destination applications, comprising:

First, Suresh discloses a system because it teaches a “computer system” used “as part of a data warehousing system.” Ex. 1004, 5:38-39, Figure 2.

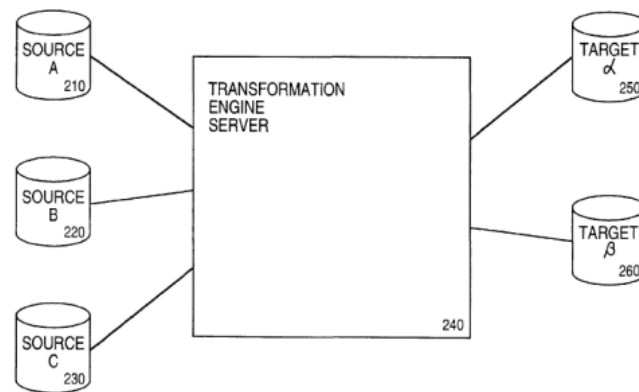


Figure 2

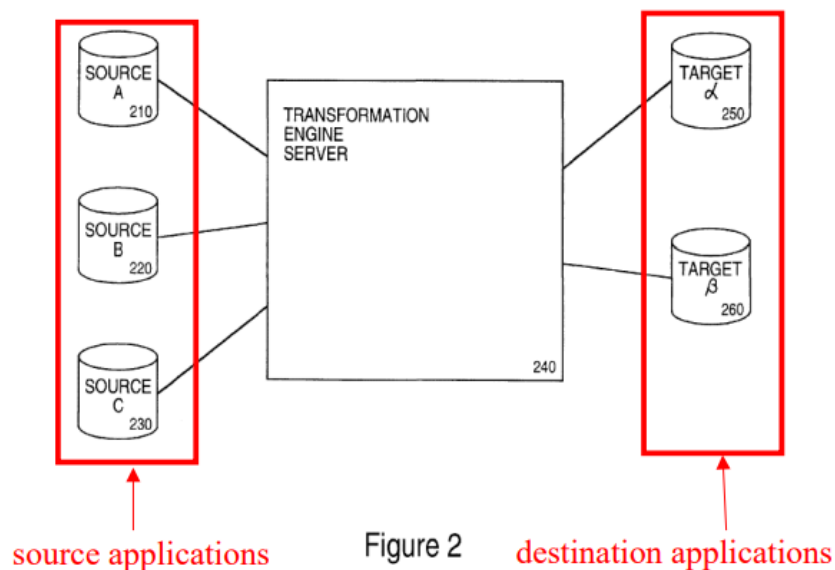
Ex. 1004, Figure 2.

Suresh also discloses a *system for integrating data* because the system includes a “repository” and a “transformation engine server.” The repository contains “mapping information relating to how data is to be mapped and transformed from target tables of the operational databases to target tables of data marts.” Ex. 1004, 6:23-29. The transformation engine server extracts and transforms data from source applications and loads the data into destination applications:

In the present embodiment of the present invention, these **operational databases provide the data source from which the transformation engine server 240 can extract, transform, then load the data into the databases 250 and 260. Databases 250 and 260 are the data warehouses or data marts that are the targets of the data ETL [Extraction/Transformation/Loading] process.**

Ex. 1004, 6:16-19 (emphasis added); Ex. 1002, ¶124-125.

Second, Suresh indicates that its system integrates data *among heterogeneous source applications and destination applications* because the transformation engine extracts data from applications, such as Source A, Source B, and Source C and loads the transformation data into target applications, such as Target α and Target β :



Ex. 1004, Fig. 2 (annotated); Ex. 1002, ¶126.

A POSITA would recognize that the *source applications* (Source A 210, Source B 220, and Source C 230) are *heterogenous* because they are different “operational databases 210, 220, and 230” that “can be any of the conventional RDMS systems (such as from Oracle, Informix, Sybase, Microsoft, etc.).” Ex. 1004, 6:8-15; Ex. 1002, ¶127. Similarly, a POSITA would recognize that the *destination applications* (Target α and Target β) are *heterogeneous* because they are different “[d]atabases 250 and 260” that are “warehouses or data marts.” Ex. 1004, 6:16-19; *see also* Ex. 1004, 2:27-28 (“there might ... many different types of data marts”); Ex. 1002, ¶128. Lastly, a POSITA would recognize that Sources A-C (*source applications*) are different (*heterogeneous*) than Targets α and β (*destination applications*) because the teachings of Suresh are how to transform data from a Source to a Target. There would be no reason to transform the data if the Source and Target were the same. Ex. 1002, ¶129.

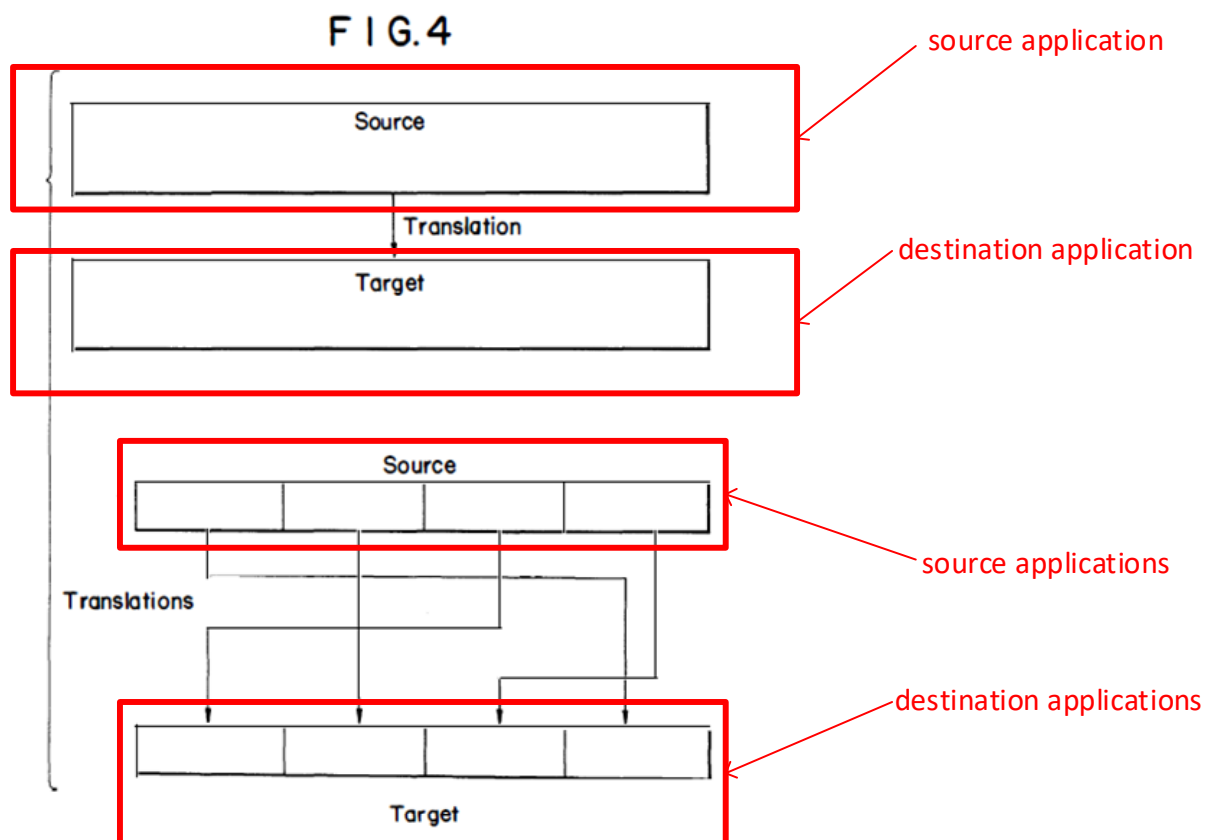
Further, a POSITA would recognize that Suresh’s Sources and Targets are *applications*.³ A POSITA would understand that Suresh’s “conventional RDMS systems” from “Oracle, Information, Sybase, Microsoft, etc.” and “data warehouses and data marts” are *applications* because an “application” is merely a

³ The ’178 patent itself uses “database application” and “database” interchangeably. *See, e.g.*, ’178 patent, Fig. 1, (element 16; “legacy database”), 4:8 (“legacy database applications 16”).

computer program designed to assist in the performance of a specific task. Ex.

1002, ¶64, *See also*. Ex. 1009). Correspondingly, Suresh's systems are software applications running on computers to perform data management tasks. Ex. 1002, ¶130.

For Ground 3, Brann describes a source application (source) and a destination application (target), see Fig. 4 of Brann annotated below.



Thus, the computer system of Suresh that extracts and transforms data from different source database systems and loads the data into different target database

systems, alone or in combination with the teachings of Brann, renders obvious “*a system for integrating data among heterogeneous source applications and destination applications.*” Ex. 1002, ¶124-135.

[1.1] *a repository of information relating to the source applications and the destination applications, the repository including a plurality of data elements each being capable of performing a discrete operation on a piece of data, and*

As discussed in [1.0], Suresh’s system includes *a repository*. See Ex. 1004, 6:23-29 (“a repository, which is used to keep track of ... mapping information”).

Suresh explains that the repository stores mapping information relating to the operational databases (*source applications*) and the target tables of data marts (*destination applications*):

A client is used to create a repository, which is used to keep track of session information as well as mapping information relating to how data is to be mapped and transformed from target tables of the operational databases to target tables of data marts.

Ex. 1004, 6:23-29.

Suresh discloses that one example of the mapping information is “ready-made transformation components.” Ex. 1004, 6:56-65. Each transformation component “is a software binary file acting as an individual unit that possesses built-in autonomy and encapsulates the functionalities of a transformation” and “is responsible for one specific type of transformation.” Ex. 1004, 5:17-20, 6:46-52.

Suresh discloses an “extensible set” of “thirteen different transformation components: source, target, expression, aggregation, filter, rank, update strategy, sequence, joiner, lookup, stored procedure, external procedure, and normalizer.” Ex. 1004, 6:56-65, 10:35-11:6.

Thus, the repository of Suresh that stores mapping information that (i) relates to how to transform data from the source to the destination and (ii) includes thirteen types of transformation components that are each responsible for a specific type of transformation renders obvious “*a repository of information relating to the source applications and the destination applications, the repository including a plurality of data elements each being capable of performing a discrete operation on a piece of data.*” Ex. 1002, ¶¶136-140.

[1.2] *[the repository including...] a first module including a plurality of instruction sets, each instruction of each instruction set activating a data element; and*

Suresh uses *a first module* because it describes that users can create a “pipeline” using a “pool” of transformation components to “build functionally more sophisticated transformations.” Ex. 1004, 6:61-65. A pipeline includes “cascading sequences of transformation components.” Ex. 1004, 7:25-27. In the example depicted in Figure 4, the cascading sequence in the pipeline includes “a source table 420, an expression transformation 422, an aggregation transformation 424, and a target table 426.” Ex. 1004, 7:3-14.

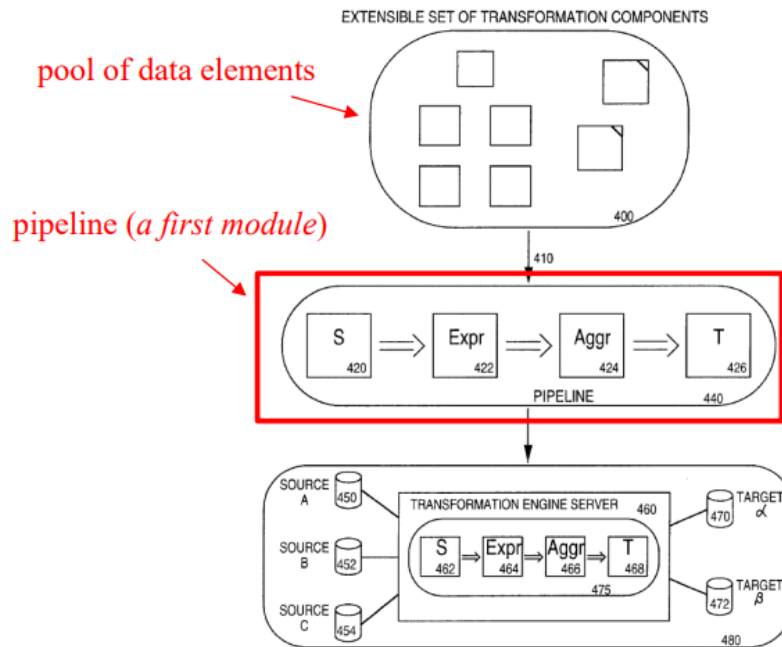
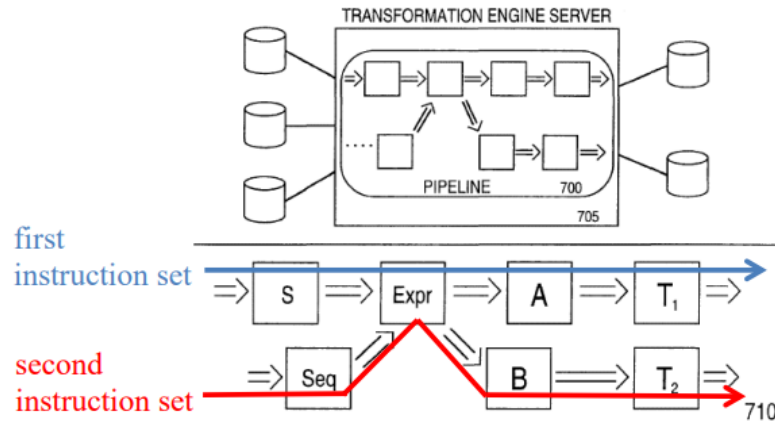


Figure 4

Ex. 1004, Fig. 4 (annotated); Ex. 1002, ¶141-142.

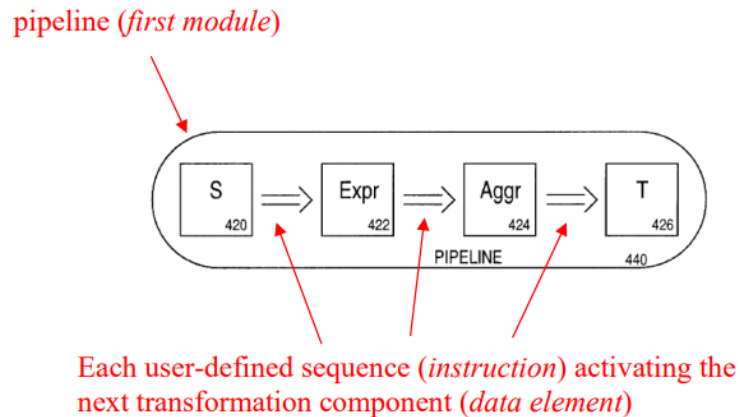
Additionally, Figure 7 depicts a pipeline with two example cascading sequences (*plurality of instruction sets*) that are configured to operate with the transformation engine server:



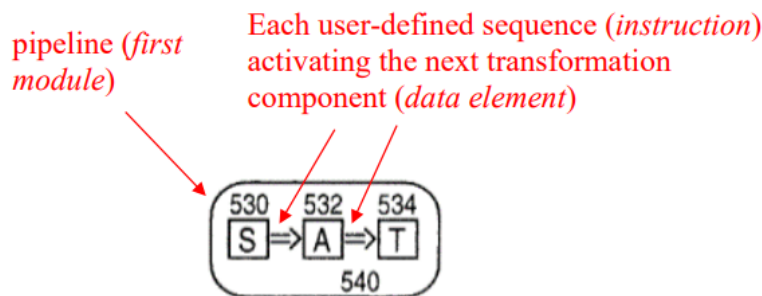
Ex. 1004, Figure 7 (in part) (annotated); Ex. 1002, ¶143-146.

Next, Suresh explains that each cascading sequence (*instruction*) in the

instruction set activates the next transformation component because the “transformation components that are being formed into the pipeline” are “coupled together (e.g. in a Lego-like fashion).” Ex. 1004, 7:38-42. The user can “form one or more sequential” pipelines, such as the pipelines depicted in Figures 4 and 5:



Ex. 1004, Fig. 4, (in part) (annotated); Ex. 1002, ¶147-148.



Ex. 1004, Fig. 5 (in part), (annotated); Ex. 1002, ¶147-148.

Suresh describes and suggests that the *repository* includes the pipeline (*first module*) because the repository keeps track of “mapping information relating to how data is to be mapped and transformed from target tables of the operational databases to target tables of data marts.” Ex. 1004, 6:23-29. A pipeline is a specific “transformation that consist of a cascading sequence of transformation

components,” which is how data is being “mapped and transformed.” Ex. 1004, 6:23-29. Accordingly, because the repository stores the information relating to how data is transformed and the pipeline represents the specific sequence for how to transform the data, a POSITA would understand that the pipeline is part of the mapping information stored by the repository. Ex. 1002, ¶149.

If Patent Owner argues that the recited “module” must not contain “data elements,” Patent Owner’s argument ignores the claim language and the other disclosures in the ’178 patent. First, the plain language of the claim recites that instruction sets are in the module but does not exclude data elements from the module. Second, the ’178 patent describes that, while the data elements may initially reside in the library, when module 70 executes, data elements 60 are “assembled into combinations of data elements (data element chains 110)” which may be in module 70. Ex. 1001, 6:36-38; *see* Figures 3-4 (depicting data elements 60 inside chain 110 and chain 110 inside module 70). Ex. 1002, ¶150.

Thus, the repository of Suresh that includes mapping information, such as pipelines having a plurality of cascading sequences, where each cascading sequence is a series of transformation components that execute in sequence in operation with the transformation engine server to add functionality renders obvious “*a repository including ... a first module including a plurality of instruction sets, each instruction of each instruction set activating a data element.*”

Ex. 1002, ¶150-151.

To the extent this claim element is not taught by Suresh, for Ground 3 it would be obvious to a POSITA to provide a first module including a plurality of instruction sets, each instruction of each instruction set activating a data element, for example as disclosed in Brann. Ex. 1002, ¶¶152-155.

Brann describes a database (*repository*) that includes data structures, for example, database tables (*first module*) that store instruction sets for activating subroutines (*data structures*). See VIII.D.1.[1.2]. The Control Program of Brann obtains a list of subroutines to be invoked and an order in which they should be invoked from the database. Ex. 1003, 7:19-30. The Control Program then invokes the required subroutines in the required order, checking that each invocation was successful. *Id.* at 12:1-9. The Control Program extracts its instructions from the database and retains them, in the form of arrays of data structures, in the computer's memory. Ex. 1003, 7:21-35.

The database (*repository*) of Brann stores a set of transformation instructions which includes the originating data elements (i.e., a field in the first structure), a resulting data element (i.e., a field in the second structure), and the Transformation Method required between them. Each one of those makes up one row in a table of the database. Ex. 1003, 10:58 – 11:10. A method table in the database stores a list of subroutines to be invoked and the order in which they should be invoked. *Id.* at

7:23-39; 11:10-23. Each instruction activates its associated subroutine. *Id.* at 7:21-35. The method table includes pointers to the particular subroutines, just as in the '178 patent. A function reference is used to activate the selected subroutine. *Id.* at 7:31-47; *See also Id.* at 11:10-19. Brann thus describes and suggests “*a first module including a plurality of instruction sets.*” Ex. 1002, ¶¶152-155.

Annotated figure 5 of Suresh illustrates how the “*first module*” suggested by Brann would be included in Suresh. The “*instruction sets*” that define the order in which the transformation components are to be executed are included in the “*first module,*” stored in the same repository as the extensible set of transformation components.

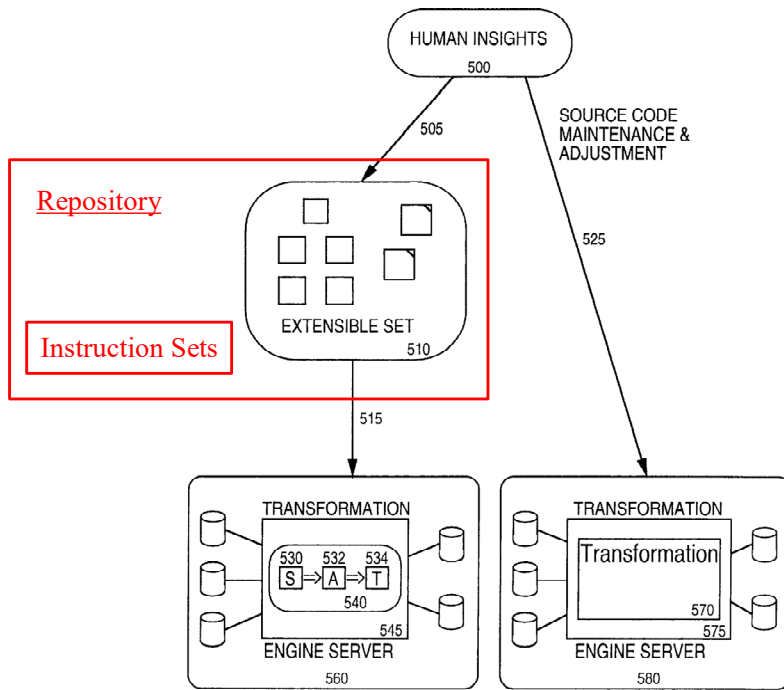


Figure 5

Ex. 1004, Fig. 5 (annotated); Ex. 1002, ¶154.

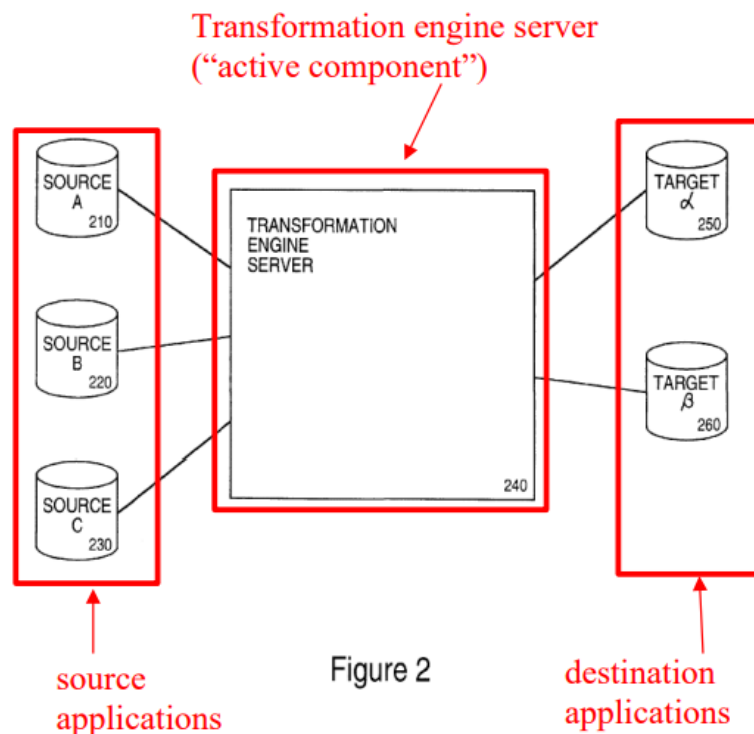
Once a pipeline, that is the sequence in which the transformation components are to be called, is created, it would be stored in the repository. A POSITA would have understood that doing so would allow the pipeline to be quickly and easily reused or even modified for different transformations. Ex. 1002, ¶¶155.

Thus, the repository of Suresh that includes mapping information, such as pipelines having a plurality of cascading sequences, where each cascading sequence defines a series of transformation components to be execute in sequence

in operation with the transformation engine server to add functionality, combined with the teachings of Brann of storing sets of transformation instructions in a database renders obvious “a repository including ... a first module including a plurality of instruction sets, each instruction of each instruction set activating a data element.” Ex. 1002, ¶136-156.

[1.3] an active component connected to the repository having an input for loading data from the source applications, and an output for populating the destination applications with processed data, the active component processing the data from the source applications by activating data elements according to the plurality of instruction sets in the module.

Suresh describes and suggests an *active component* because it describes a “transformation engine server” that “can extract, transform, then load the data.” Ex. 1004, 6:15-20.



Ex. 1004, Fig. 2 (annotated); Ex. 1002, ¶157.

A POSITA would understand that Suresh describes and suggests that the transformation engine server is connected to the repository because the transformation engine server accesses the pipeline (and its transformation components) from the repository and then executes the pipeline as a pipelined transformation. Ex. 1002, ¶158.

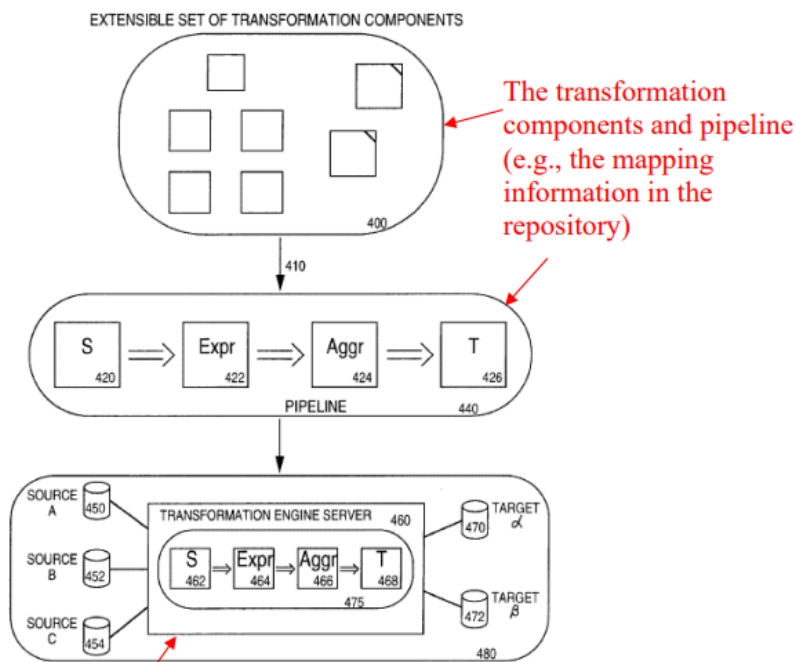


Figure 4

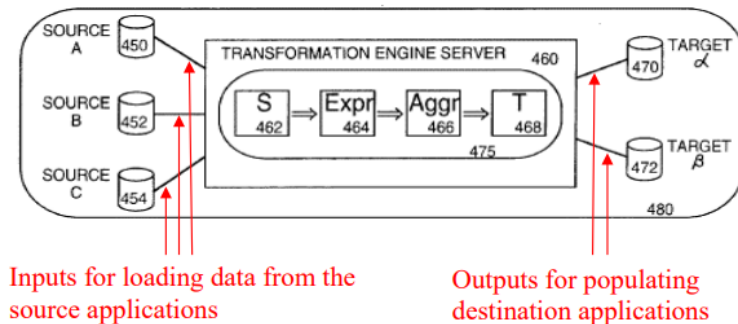
The pipeline and transformation components (e.g., the mapping information from the repository) being executed by the transformation engine server as a pipelined transformation

Ex. 1004, Fig. 4 (annotated); Ex. 1002, ¶158-159.

To the extent Suresh does not include a figure depicting the repository connected to the transformation engine server, a POSITA would have known that the repository and transformation engine would be connected because the

information in the repository needs transferred to the transformation engine to perform the transformation. Ex. 1002, ¶161, citing Ex. 1010.

Suresh describes that the transformation engine has *an input for loading data from the source applications, and an output for populating the destination applications with processed data*, because “[d]ata from any of the source tables 450, 452, and 454 can be transformed by the pipelined transformations within transformation engine server 460 and stored in either and/or both of target tables 470 and 472.” Ex. 1004, 7:27-32, *see also id.*, 6:27-29 (“data is to be mapped and transformed from **target tables of the operational databases to target tables of data marts.**”). Suresh’s Figure 4 illustrates the inputs for loading and the outputs for populating:



Ex. 1004, Fig. 4 (annotated); Ex. 1002, ¶162.

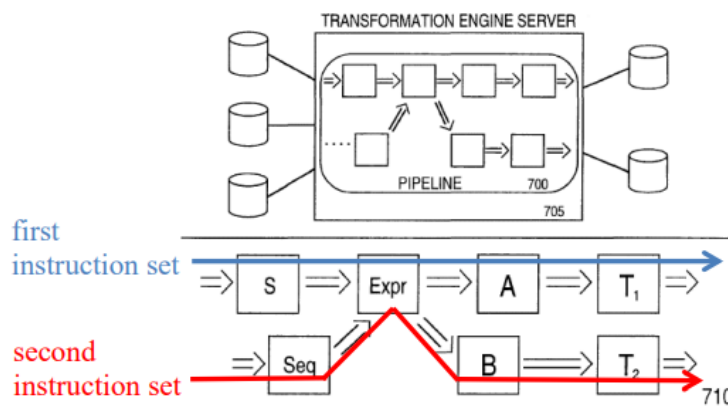
Suresh describes *the active component processing the data from the source applications by activating data elements according to the plurality of instruction sets in the module* because the transformation engine server (*active component*) transforms data from source applications by activating the sequence of

transformation components (*activating data elements according to plurality of instruction sets*) in the pipeline (*first module*) as a pipelined transformation:

The pipelined transformations exist within the transformation engine server 460. Data from any of the source tables 450, 452, and 454 can be transformed by the pipelined transformations within transformation engine server 460 and stored in either and/or both of target tables 470 and 472.

Ex. 1004, 7:27-32.

The transformation engine server processes data from source applications by activating different transformation components according to the cascading sequences in the pipeline. Ex. 1004, 9:13-38 (“the transformation engine server automatically starts to identify the individual pipeline(s) so that it can execute the pipeline(s)”):



Ex. 1004, Fig. 7 (in part) (annotated); Ex. 1002, ¶163-164.

Thus, the transformation server of Suresh that (i) is connected to the

repository of transformation components, (ii) has an input for loading data from source applications, (iii) has an output for populating data in target applications, and (iv) transforms the source's data by activating transformation components according to cascading sequences in the pipeline renders obvious “*an active component connected to the repository having an input for loading data from the source applications, and an output for populating the destination applications with processed data, the active component processing the data from the source applications by activating data elements according to the plurality of instruction sets in the module.*” Ex. 1002, ¶¶157-164.

3. Claim 2

[2.1] *The system of claim 1 wherein the plurality of instruction sets are predefined to facilitate interface with a particular destination application.*

Suresh discloses the cascading sequences of transformation components (*plurality of instruction sets*) are defined to facilitate interface with a particular target table of a data mart (*a particular destination application*) because “data is to be mapped and transformed from target tables of the operational databases to target tables of data marts.” Ex. 1004, 6:26-29; *see also* Ex. 1004, 7:9- 32 (“[d]ata from any of the source tables 450, 452, and 454 can be transformed by the pipelined transformations within transformation engine server 460 and stored in either and/or both of target tables 470 and 472”).

Second, Suresh discloses that the cascading sequences (*plurality of*

instruction sets) are *predefined* because users define the pipelines before the transformation engine service performs the pipelined transformation. Ex. 1004, 7:9- 11 (“the step taken by the users ... to manually form a new transformation”), 7:25-27; *see also* Ex. 1004, 7:27-32 (“Data ... can be transformed by the pipelined transformations within transformation engine server 460”). Ex. 1002, ¶¶166-167.

Thus, the cascading sequences of Suresh that (i) interface with the target table in a data mart and (ii) are defined by users and are stored in the repository before the pipelined transformation executes renders obvious the recited limitation. Ex. 1002, ¶¶165-168.

4. Claim 3

[3.1] *The system of claim 1 further comprising a second module including a plurality of instruction sets, each instruction of each instruction set activating a data element,*

First, as discussed in [1.2], Suresh discloses a pipeline that includes cascading sequences. The pipeline activates the next transformation component in the sequence.

Second, Suresh discloses multiple pipelines. Ex. 1004, 8:5-9. Any of the pipelines, such as pipeline 630, can be a second pipeline (*second module*) with cascading sequences (*plurality of instruction sets*):

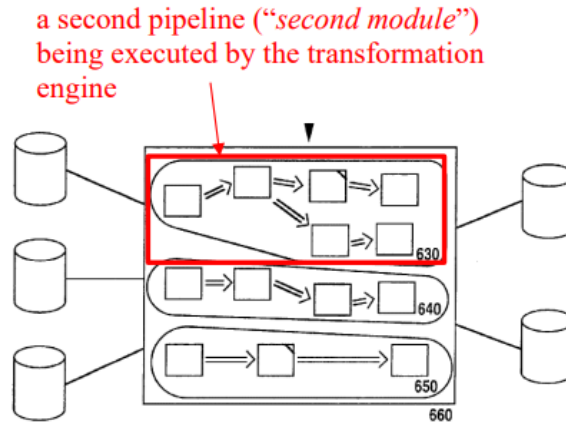


Figure 6

Ex. 1004, Fig. 6 (annotated); Ex. 1002, ¶171.

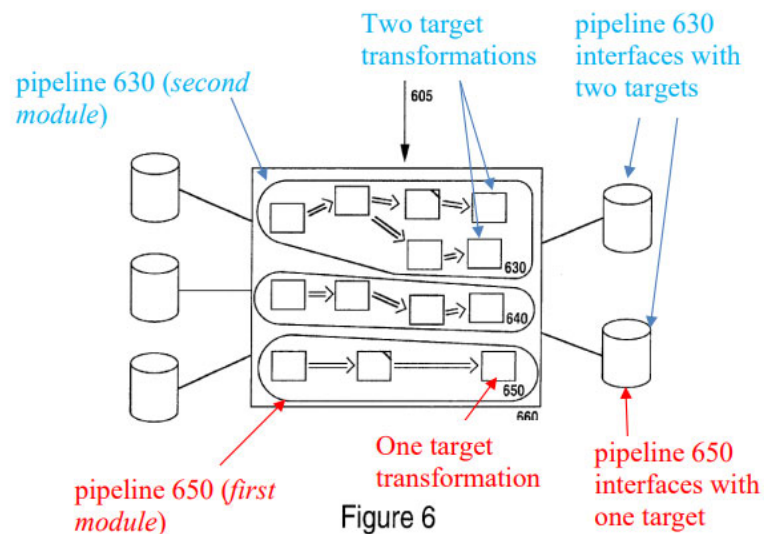
Brann describes the sequences in which the transformation components are activated, that is the “pipeline(s),” may include a multiplicity of sets of instructions. See [1.2] The second pipeline of Suresh would have been modified in the same way as the first pipeline. *Id.* A POSTIA would have combined this teaching of Brann with Suresh. *Id.* Thus, a second pipeline of Suresh that includes cascading sequences of instructions to activate transformation components renders obvious the recited limitation, alone or in view of Brann. Ex. 1002, ¶¶169-172.

[3.2] *the first module plurality of instruction sets being defined for interface with a first group of destination applications, the second module plurality of instruction sets being defined for interface with a second group of destination applications.*

Suresh discloses each pipeline (*first module and second module*) can push data to different targets (*a first group of destination applications and a second group of destination applications*) because each pipeline can output “to one or

more subsequent ... targets.” Ex. 1004, 8:31-35; *see also id.*, 7:27-37 (outputting to “either and/or both” of targets 470 and 472). The pipelines each interface with the target(s) using a “target transformation” that “receive[s] data from other transformations” and “then make[s] the data available to the target.” Ex. 1004, 10:41-44, Fig. 7 (depicting target transformation T1 and T2 pushing data to the targets). Ex. 1002, ¶173.

Suresh’s Figure 6, for example, illustrates pipeline 650 (*first module*) uses one target transformation to interface with one target (*first group of destination applications*) and pipeline 630 (*second module*) uses two target transformations to interface with two targets (*second group of destination applications*):

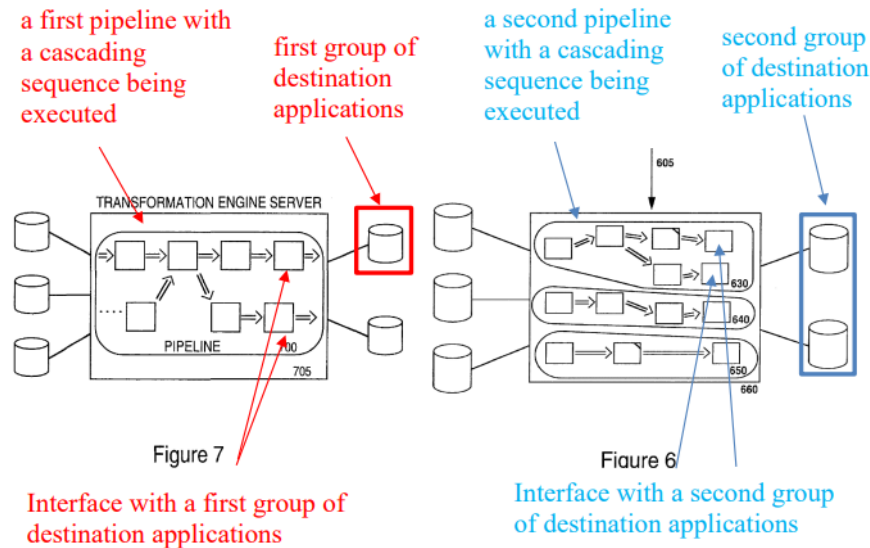


Ex. 1004, Fig. 6 (annotated); Ex. 1002, ¶174.

Because Suresh discloses data from a pipeline can be stored in one or more targets (Ex. 1004, 8:31-35), a POSITA would have found it obvious to implement

different pipelines that each interface with different groups of one or more targets.

This embodiment is depicted below:



Ex. 1004, Figs. 6 and 7 (in part) (annotated); Ex. 1002, ¶175.

Thus, Suresh discloses cascading sequences in the first pipeline that interface with one target and cascading sequences in the second pipeline that interface with two targets, which renders obvious the recited limitation. Ex. 1002, ¶¶169-176.

5. Claim 4

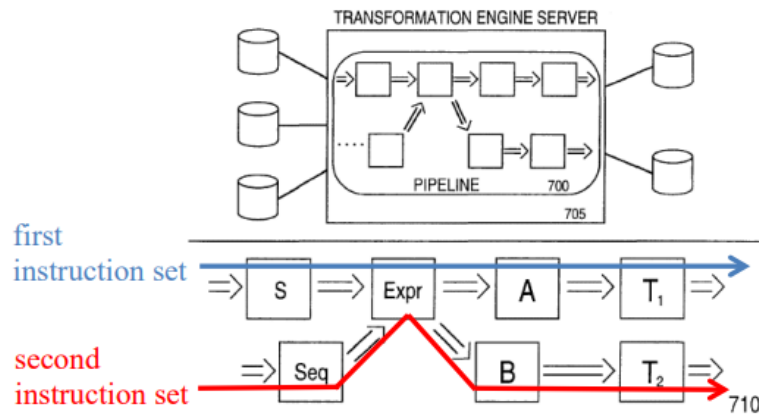
[4.1] The system of claim 3 wherein at least one data element of the plurality of data elements is activated by both the first module plurality of instruction sets and the second module plurality of instruction sets.

As discussed in [1.1], Suresh discloses an extensible set of ready-made transformation components (*data elements*). Ex. 1004, 6:56-65.

Suresh discloses users can create “pipelined transformations,” such as pipeline 700 and 630 (*first module and second module*) from the transformation

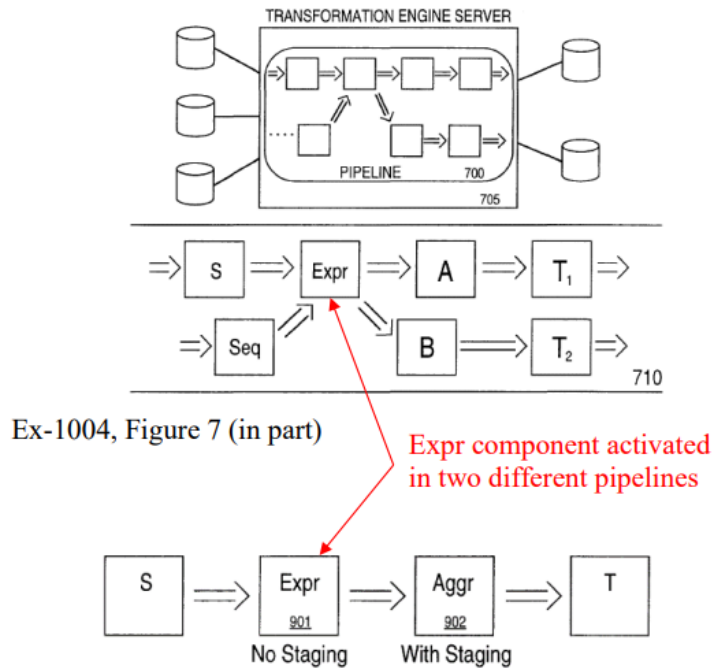
components (*data elements*). Ex. 1004, 6:56-65; *see also Id.*, 7:35-37.

Suresh also discloses the ready-made transformation components are used by multiple instruction sets, as exemplified by the “Expr” component in Figure 7:



Ex. 1004, Fig. 7 (in part) (annotated); Ex. 1002, ¶177-181.

Suresh further discloses different pipelines can each activate a common transformation component, particularly since the components are being selected from a “pool.” Ex. 1004, 6:56-65 (“transformation components can be thought of as a pool [] i.e., a base of ready-made transformation components from which the users can draw on.”). For example, the “Expr” component is used in the example pipeline in Figure 7 and in a different pipeline in Figure 9:



Ex-1004, Figure 7 (in part)

Ex. 1004, Fig. 9 (in part); Ex. 1002, ¶182.

The selected transformation component would be *activated by both the first module plurality of instruction sets and the second module plurality of instruction sets* as discussed in [1.3] and [3.1]. Specifically, Suresh discloses the transformation component would be *activated* during the transformation process.

Thus, at least one transformation component of Suresh that (i) is part of a ready-made set of transformation components, (ii) can be included in different pipelines, and (iii) is activated during the pipelined transformations renders obvious the recited limitation. Ex. 1002, ¶¶177-184.

6. Claim 5

[5.1] The system of claim 3 wherein the repository further includes configuration data storage for storing configuration data describing data in the source

applications, the configuration data storage including a first portion used by the first module and a second portion used by the second module.

As discussed in [1.1], Suresh discloses the repository contains “mapping information relating to how data is to be mapped and transformed from target tables of the operational databases to target tables of data marts.” Ex. 1004, 6:23-29.

Suresh describes and suggests the repository stores *configuration data describing data in the source applications* because the repository stores mapping information “relating to how data is to be mapped and transformed from target tables of the operational databases to target tables of data marts.” Ex. 1004, 6:23-29; *see also id.*, 7:8-19. Both the information for how data is to be mapped and transformed and the source table that stores the original data teach the *configuration data describing data in the source applications*. Ex. 1004, 7:66-8:11; Ex. 1002, ¶185-187.

The portion of the repository storing the mapping information, including how the data is to be mapped and transformed and the source table, is the *configuration data storage*. Ex. 1002, ¶88; See Ex. 1004, 6:15-29.

As discussed in [3.1]-[3.2], Suresh discloses multiple pipelines (*first module and second module*). It would have been obvious to a POSITA that the mapping information of the first pipeline, e.g., pipeline 700 (*first module*), would be stored in a *first portion* of the *configuration data storage* and that the mapping

information of the second pipeline, e.g., pipeline 630 (*second module*) is stored in a *second portion* of the *configuration data storage*. Storing mapping information for each of the pipelines allows the transformation engine server to perform “pipelined transformations” by different pipelines in parallel. Ex. 1004, 8:12-14; *see also* Ex. 1004, 7:35-37 (“multiple processes can run simultaneously within parallel pipeline stages”).

Accordingly, the repository of Suresh that (i) stores mapping information describing data in tables of the source databases and (ii) facilitates executing pipelined transformations in parallel, dedicated threads by storing the mapping information for a first pipeline and a second pipeline, renders obvious the recited limitation. Ex. 1002, ¶¶185-189.

7. Claim 7

[7.1] *The system of claim 1 wherein the repository is capable of receiving a plurality of modules.*

As discussed in [1.1], Suresh discloses a repository, and the repository contains “transformation components available for use to build functionally more sophisticated transformations.” Ex. 1004, 7:5-7:8. Users “manually form a new transformation that consists of cascading sequence of transformation components.” Ex. 1004, 7:9-12, 8:5-9, Figures 4, 6, 7. Suresh describes and suggests that the repository stores mapping information, and the pipelines are the mapping information that describes how the data is transformed. See Ex. 1004, 6:25-29,

7:10-11.

Accordingly, a POSITA would have recognized, since the repository stores mapping information (e.g., pipelines), the repository would receive the user-made pipelines as the pipelines are created as the pipeline would be stored to execute the transformation. Ex. 1002, ¶¶191-194.

Moreover, Brann describes the database stores a plurality of translation programs which include the sequence in which data elements are activated. *See* VIII.D.1.[1.2]. A POSITA would understand that the sequences in which the transformation components to be called in Suresh would be stored in the repository. Ex. 1002, ¶¶195-197.

A POSITA would have considered the teachings of Brann and Suresh because they are analogous prior art in the same field of endeavor as the '178 patent – integrating data among multiple, disparate sources and targets. Ex. 1002, ¶196.

Suresh would have benefited from Brann's teachings. A POSITA would have recognized that the database of Brann could be used to store the subroutines used for the translation, along with the translation instructions, data, and the like. Ex. 1002, ¶197 There would have been a reasonable expectation of success in the combination as storing transformation instructions was well-known Ex. 1002, ____

Thus, Suresh alone, or in view of Brann renders obvious the recited

limitation. Ex. 1002, ¶¶191-198.

8. Claim 9

[9.1] *The system of claim 1 wherein the active component is capable of loading entire files from a source application.*

Suresh describes and suggests that the “transformation engine server 240 [active component] can extract ... data” from a data source (*source application*). Ex. 1004, 6:15-20. Suresh provides an example in which the data from the source and provided to the destination is a “flat file.” Ex. 1004, 10:39-41 (“**flat files** that provide data for the data mart/data warehouse”).

A POSITA would have understood that the flat files for the source transformation component would be loaded from one of the data sources, such as Sources A-C, for subsequent transformation by the pipeline. *See, e.g.*, Ex. 1014, 10:56-58 (describing “Source Data” originating from an “ASCII file”).

Accordingly, the transformation engine server of Suresh that loads flat files from a data source with a source transformation component renders obvious the recited limitation. Ex. 1002, ¶¶199-201.

9. Claim 10

[10.1] *The system of claim 1 wherein the repository further includes a library for storing the plurality of data elements.*

As discussed in [1.1], Suresh describes and suggests a repository storing session information and mapping information. *See* Ex. 1004, 6:23-29. Suresh further discloses the mapping information includes a pool (*library*) of ready-made

transformation components (*data elements*) that are available to build transformations. Ex. 1004, 6:56-65 (“transformation components can be thought of as a pool [] i.e., a base of ready-made transformation components from which the users can draw on.”).

A POSITA would have recognized that a portion of the repository that stores a pool of ready-made transformation components is a *library*. Ex. 1002, ¶203-204.

Accordingly, the repository of Suresh that stores a pool of pre-built transformation components renders obvious the recited limitation. Ex. 1002, ¶¶202-205.

10. Claim 23

[23.0] *A system for integrating data among heterogeneous source applications and destination applications, comprising:*

See [1.0].

[23.1] *configuration data storage for storing configuration data relating to the source applications;*

See [5.1].

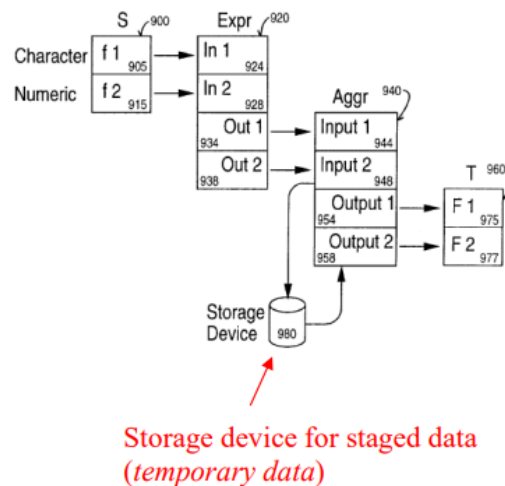
[23.2] *temporary data storage for storing temporary data being integrated among the source and the destination applications;*

As discussed in [1.3], Suresh describes and suggests that the transformation engine extracts and transforms data from source applications and loads the data into destination applications. Ex. 1004, 6:16-19, 7:27-31.

Suresh describes and suggests *temporary data storage* because it describes

and suggests that certain transformation components require “data staging as it transform incoming data fields.” The data “staging occurs in RAM, then rims out to a storage disk as RAM storage space runs out.” Ex. 1004, 10:13-34.

Suresh’s Figure 9 depicts the RAM and storage disk memory (“*temporary data storage*”) that temporarily stores staged data during the pipelined transformation:



Ex. 1004, Fig. 9, in part (annotated); Ex. 1002, ¶¶209-210.

Suresh further discloses multiple transformation components have data staging capability. Ex. 1004, 10:64-11:1 (“source, target, aggregation, rank, and joiner transformations are all staged transformations. The data generated by these transformations are automatically staged”).

Thus, the RAM and storage disk memory of Suresh that stores staged data as the data is transformed from the source databases to the target databases using pipelined transformations renders obvious “*temporary data storage for storing*

temporary data being integrated among the source and the destination

applications.” Ex. 1002, ¶¶208-213.

[23.3] *a library of data elements, each data element performing a predefined function;*

As discussed in [10.1], Suresh discloses a pool of “ready-made transformation components” (*a library of data elements*). Ex. 1004, 6:56-65.

Suresh describes and suggests “*each data element performing a predefined function*” because Suresh discloses thirteen components, each having a predefined transformation (*predefined function*), including “source, target, expression, aggregation, filter, rank, update strategy, sequence, joiner, lookup, stored procedure, external procedure, and normalizer.” Ex. 1004, 10:35-11:6; *see also id.*, 5:17-21.

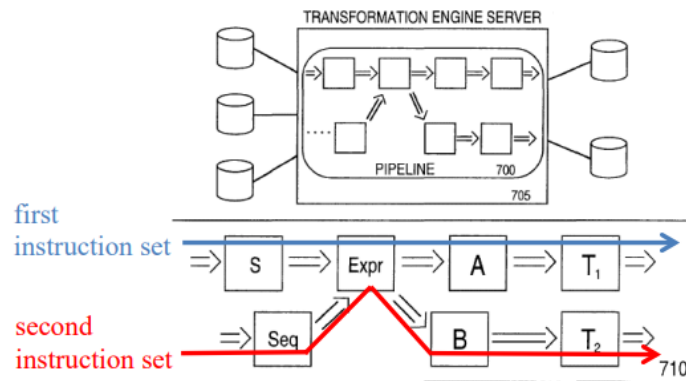
Accordingly, a pool of thirteen ready-made transformation components, each performing a different transformation renders obvious “*a library of data elements, each data element performing a predefined function.*” Ex. 1002, ¶¶214-215.

[23.4] *a first module including a plurality of instruction sets, each instruction set specifying a plurality of data elements to form a chain of data elements; and*

As discussed in [1.2], Suresh and/or Suresh in view of Brann describes and suggests a *first module including a plurality of instruction sets*. See VIII.E.1.-2.

Suresh’s Figure 7 depicts two example cascading sequences (*plurality of*

instruction sets):



Ex. 1004, Fig. 7 (in part) (annotated); Ex. 1002, ¶216.

As discussed in [1.2], Suresh discloses each cascading sequence (*instruction set*) specifies transformation components (*a plurality of data elements*). Moreover, because the transformation components are “coupled together (e.g. in a Lego-like fashion),” they *form a chain of data elements*. Ex. 1004, 7:38-42. For example, the first instruction set of Figure 7 forms a chain of transformation components “S → Expr → A → T₁,” and the second instruction set forms “Seq → Expr → B → T₂.”

Thus, a pipeline having a plurality of cascading sequences, where each cascading sequence is a series of transformation components coupled together renders obvious “*a first module including a plurality of instruction sets, each instruction set specifying a plurality of data elements to form a chain of data elements.*” Ex. 1002, ¶¶216-218.

To the extent this claim element is not taught by Suresh, it would be obvious to a POSITA to provide a first module including a plurality of instruction sets, each

instruction of each instruction set activating a data element, for example as taught in Brann.

Brann teaches a list of subroutines to be invoked and an order in which they should be invoked. Ex. 1003, 7:24-30. The Control Program then invokes the required subroutines in the required order, checking that each invocation was successful. *Id.* at 12:1-9. The Control Program extracts its instructions from the database and retains them, in the form of arrays of data structures, in the computer's memory. Ex. 1003, 7:21-35. *See* VIII.E.1.-2.

Thus, Suresh teaching pipelines having a plurality of cascading sequences, where each cascading sequence defines a series of transformation components to be execute in sequence in operation with the transformation engine server to add functionality combined with the teachings of Brann of storing sets of transformation instructions in a database renders obvious “*a first module including a plurality of instruction sets, each instruction set specifying a plurality of data elements to form a chain of data elements.*” Ex. 1002, ¶¶219-221.

[23.5] *an active component operably associated with the configuration data storage, the temporary data storage, the library, and the first module,*

As discussed in [5.1] and [10.1], Suresh discloses that the *configuration data storage* and the *library* are included in the repository because the repository tracks the mapping information relating to how the data is transformed. Ex. 1004, 6:23-29. As discussed in [1.3], the transformation engine server (“*active component*”) is

connected to the repository.

Further, as discussed in [23.2], the transformation engine is connected to the RAM and storage disk for storing staged data as the transformation engine executes the pipelined transformation.

Accordingly, because the transformation engine server is (i) connected to the repository (which includes the configuration data storage and the library), (ii) connected to the RAM for storing staged data, and (iii) executes the pipeline as a pipelined transformation, Suresh renders obvious “*an active component operably associated with the configuration data storage, the temporary data storage, the library, and the first module.*” Ex. 1002, ¶¶222-224.

[23.6] *the active component including a loader for loading data from the source applications to the temporary data storage according to the configuration data relating to the source applications, and*

Suresh describes and suggests *a loader for loading data from the source applications* because it teaches a source transformation component (*loader*) that “extracts from the operational database” data during the ETL process. Ex. 1004, 9:59-62; *see also* 10:39-41.

Suresh describes and suggests that the *loader* loads the data into the *temporary data storage* because, as discussed in [23.2], data in the source transformation component is “staged” in the RAM and storage disk memory (*temporary data storage*).

Suresh describes and suggests that the data is loaded *according to the configuration data relating to the source applications*, because, as discussed in [5.1], the mapping information (*configuration data*) describes how data is “mapped and transformed from target tables of the operational databases.” Ex. 1004, 6:23-29; *see also id.*, 7:8-19 (“a pipeline can include a source table 420 ... [and the] original data, in the form of records, are stored in source table 420.”).

Suresh describes and suggests that the transformation engine server (*active component*) includes the source transformation component (*loader*) because the source transformation component is in the pipeline used by transformation engine server to perform pipelined transformations. Ex. 1004, 7:27-31.

Accordingly, the source transformation component that (i) extracts data from the operational database, (ii) then loads the data into the RAM or storage disk memory, (iii) according to the mapping information, and (iv) is included in the pipeline during the pipelined transformation in the transformation engine server renders obvious “*the active component including a loader for loading data from the source applications to the temporary data storage according to the configuration data relating to the source applications.*” Ex. 1002, ¶¶225-229.

[23.7] *[the active component including ...] an interfacier for populating the destination applications with data from the temporary data storage after the data is processed by the active component,*

Suresh describes and suggests *an interfacier for populating the destination*

applications with data ... after the data is processed by the active component because it describes and suggests a target transformation component (*interfacer*) that stores the transformed data into data marts after the transformation engine server transforms the data. Ex. 1004, 10:42-44 (“[t]he target transformation maintains database objects or files that receive data from other transformations. These targets then make the data available to data mart users for decision support”).

Suresh describes and suggests the target transformation component (*interfacer*) populates the targets with *data from the temporary data storage* because the target transformation component stages data during the transformation. Ex. 1004, 10:64-11:1, 10:30-31.

Suresh describes and suggests the transformation engine server (*active component*) includes the target transformation component (*interfacer*) because the target transformation component is in the pipeline used by transformation engine server to perform pipelined transformations. Ex. 1004, 7:27-31.

Accordingly, the target transformation component that (i) stages data in temporary memory, (ii) then loads the transformed data into the data marts, and (iii) is included in the pipeline during the pipelined transformation in the transformation engine server renders obvious “[*the active component including ... an interfacer for populating the destination applications with data from the*”

temporary data storage after the data is processed by the active component.” Ex.

1002, ¶¶230-233.

[23.8] *the active component processing the data by communicating with the first module and activating data element chains according to instruction sets of the first module.*

See [1.3].

11. Claim 24

[24.1] *The system of claim 23 wherein the plurality of instruction sets of the first module are configured to operate with a particular destination application.*

See [2.1].

12. Claim 25

[25.1] *The system of claim 23 further comprising a second module including a plurality of instruction sets specifying chains of data elements,*

See [3.1].

[25.2] *at least one of the data elements specified by the instruction sets of the first module also being specified by the instruction sets of the second module.*

See [3.2].

F. Grounds 4 and 5: Claims 8 and 19 are obvious under 35 U.S.C. §103(a) over Suresh in view Beauchamp or Suresh in view of Brann and Beauchamp

Grounds 4 and 5 are discussed together below. Brann is used to supplement Suresh for elements 1.0 and 1.2.

1. Summary of Beauchamp

Beauchamp discloses “computer tools for designing and implementing

integrated enterprise processes on computer systems.” Ex. 1006, 1:11-14. One tool in Beauchamp is the “universal client,” which “provides an interface for ERP systems, database applications, best-of-breed solutions, desktop applications, legacy systems, and even existing enterprise application integration solutions.” Ex. 1006, 9:8-12. Beauchamp further discloses that the “set of standard screens employed through the universal client may be used as building blocks to construct any business process.” Ex. 1006, 9:25-28.

2. Reasons to Combine Suresh with Beauchamp, and Brann

A POSITA would have considered the teachings of Suresh and Brann and Beauchamp because they are analogous prior art in the same field of endeavor as the ’178 patent – integrating data among multiple applications, including databases. Ex. 1002, ¶240-245. The combination of Suresh and Brann is discussed in VIII.E.1.

Suresh would have benefited from Beauchamp’s teachings in several ways.

a) Storing data in databases of ERP products

Suresh (Suresh combined with Brann) would have benefited from Beauchamp’s teachings, that the data transformation among heterogenous systems of Suresh could be used with the example ERP system as disclosed Beauchamp. A POSITA would have understood that the ETL process of Suresh could be applied to different types of systems. Ex. 1004, 6:8-15. Beauchamp provides a list of

known types of systems, including an ERP system and a database system, among others, that needed integration. Ex. 1006, 9:8-11, Figure 11. And, it was also well-known that an ERP system included databases. Ex. 1011; Ex. 1006, Figure 11. A POSITA would have known that one example of the data mart targets of Suresh would be a database/application in an ERP system, as taught by Beauchamp. *See also* Ex. 1012, 13:4-20 (discussing an integration system for integrating data among ERP applications, data warehouses, and database management systems). Ex. 1002, ¶¶243-250. There would have been a reasonable expectation of success in the combination because Suresh integrates data among different databases, and Beauchamp's ERP system is just another system that includes a database. Ex. 1002, ¶¶250-251.

b) Validating data during transformation

It was well-known that data in operational databases is dirty, e.g., undefined values for invalid records, misspelled names, and missing fields. Ex. 1004, 1:35-38. As such, it is desirable to correct data when it is extracted from a database.

A POSITA, knowing that data in operational databases is often dirty, would have looked to art that describes techniques for validating data. Beauchamp describes validation based on “validation rules” and provides details for using validation rules to correct errors. Ex. 1006, 21:27-28, 16:60-65.

Using validation rules as taught in Beauchamp would provide a flexible

approach for validating different data from different operational databases of Suresh to correct invalid, null or of range data values, and format the data.

Accordingly, there would have been a reasonable expectation of success in the combination because different data validation techniques were already well-known and applied to validate data. *See* Ex. 1014, 1:31-36, 5:30-38, 17:1-5; *see* Ex. 1012, 19:50-53 (“[t]he system 100 validates every message produced by an adapter 621 or transformer 629 using the validation criteria”). Ex. 1002, ¶¶252-257.

3. Claim 8

[8.1] *The system of claim 1, wherein the destination applications are included within an ERP product.*

Suresh, alone or in view of Brann, in view of Beauchamp render this limitation obvious.

As discussed in [1.0], Suresh discloses a system that integrates data between heterogeneous source and target database applications.

Beauchamp describes and suggests it was desirable for data integration systems to integrate all applications, including ERP systems and other databases. Ex. 1006, 9:8-11 (“provide a single business process interface for ERP systems, database applications”), 19:29-34.

Considering that both Suresh and Beauchamp teach data integration for different applications, including databases and data applications, it would have been obvious to a POSITA that the data transformation among heterogenous

systems of Suresh could be used with the example ERP system as disclosed

Beauchamp. Ex. 1002, ¶¶258-262.

Accordingly, the data transformation to destination databases of Suresh, alone or in view of Brann, combined with the technique of Beauchamp of providing transformed data to an ERP system render obvious the recited limitation.

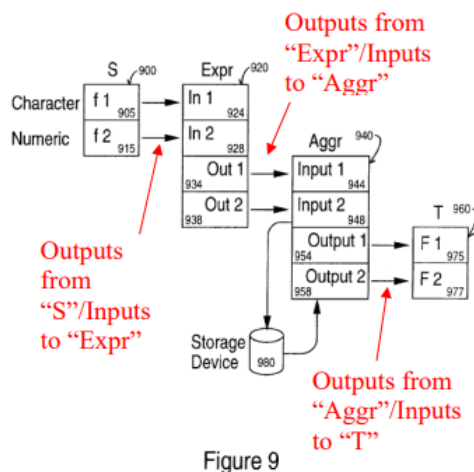
Ex. 1002, ¶¶258-263.

4. Claim 19

[19.1] *The system of claim 1, wherein each of the plurality of data elements includes related inputs, related outputs, mapping rules, and validation rules.*

Suresh, alone or in view of Brann, and Beauchamp render this limitation obvious.

Suresh discloses each transformation component (*each of the plurality of data elements*) has “one or more sets of inputs” and “can output one or more data sets.” Ex. 1004, 8:27-34.



Ex. 1004, Fig. 9 (annotated); Ex. 1002, ¶266.

Beyond the depictions in Figure 9, the source transformation component (“S”) includes *inputs* because “source **900** extracts from the source operational database two fields.” Ex. 1004, 9:59-60. Similarly, the target transformation component (“T”) includes *outputs* because “target transformation maintains database objects or files that receive data from other transformations” and “then make the data available to data mart users.” Ex. 1004, 7:14-15, 10:41-44.

Suresh also describes and suggests each transformation component (*data element*) includes *mapping rules*. Ex. 1004, 9:20-23. Each of the thirteen different transformation components has its own mapping rules. Ex. 1004, 10:35- 52, 7:3-25.

It would have been obvious to a POSITA that the transformation engine server in Suresh would include rules to validate data at each transformation component. For example, Suresh discloses an example aggregation transformation component has an output 2 that represents the “the sum of all row entries from input 2.” Ex. 1004, 10:21-26. It would have been obvious to a POSITA to validate that the data in output 2 is mapped to a numeric field (and not a character field) when a user manually selects components for inclusion into a pipeline (or other numerical validation). Ex. 1004, 5:38-39.

In addition to Suresh’s teaching of validating output, Beauchamp describes

and suggests validating data according to validation rules. Ex. 1006, 21:27-28.

Beauchamp further describes and suggests that validation can be performed “at the beginning or end of a []process.” Ex. 1006, 16:62-65. In a combination, it would have been obvious to a POSITA to use Beauchamp’s validation rules with Suresh to ensure that the data remains valid throughout the transformation process. Ex. 1002, ¶253-257. Validating data would be advantageous because it would clean empty, invalid, null and out-of-range values. See Ex. 1006, 2:66-67. Using validation rules while integrating data was well-known in the art. See Ex. 1012, 19:51-52, 30:60-63.

Thus, a combination of Suresh and Beauchamp describes and suggests data elements that include inputs, outputs, mapping rules and validation rules, which renders obvious the recited limitation. Ex. 1002, ¶¶267-272.

IX. DISCRETIONARY DENIAL WOULD BE INAPPROPRIATE

A. Discretionary denial under the *Fintiv* factors is not appropriate

Time-to-trial is not to be a dominating factor in the *Fintiv* analysis in view of the Director’s recent guidance on *Fintiv*. See Memorandum for the Interim Procedure for Discretionary Denials in AIA Post-Grant Proceedings with Parallel District Court Litigation, at 8 (June 21, 2022). Instead, a “compelling unpatentability challenge” is a more important consideration. *Id.* at 4. For the reasons set forth above, this case presents a compelling unpatentability challenge.

Moreover, the six factors considered for §314 denial strongly favor institution. *See Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (PTAB Mar. 20, 2020) (precedential). The district court case is at an early stage. Amended pleadings may be filed until July 2024. Ex. 1013, 4. Further, a hearing on claim construction is not scheduled until September 2024 and trial is currently scheduled for March 2025. *Id.* at 1, 3. Petitioner has diligently prepared and filed this petition after being served Patent Owner’s infringement contentions. *Id.* at 4. The petition is also well within the one-year timeframe allowed by Congress.

1. No evidence regarding a stay

No motion to stay has been filed, so the Board should not infer the outcome of such a motion. *Sand Revolution II LLC v. Continental Intermodal Group – Trucking LLC*, IPR2019-01393, Paper 24 at 7 (PTAB June 16, 2020) (informative); *see also Dish Network L.L.C. v. Broadband iTV, Inc.*, IPR2020-01359, Paper 15 at 11 (PTAB Feb. 12, 2021) (“It would be improper to speculate, at this stage, what the Texas court might do regarding a motion to stay...”). Thus, this factor is neutral on discretionary denial.

2. Parallel proceeding trial date

The district court has set a tentative trial date of March 2025. Ex. 1013, 1. Thus, this factor is neutral on discretionary denial.

3. Investment in the parallel proceeding

The co-pending litigation is in its early stages, and the investment in it has

been minimal. Fact discovery has just begun and claim construction and expert discovery have not begun. *See* Ex. 1013, 3, 4. Petitioner learned which claims were being asserted September 14, 2023, and served preliminary invalidity contentions on November 9, 2023 *See* Ex. 1013, 4 (infringement contentions served on September 14, 2023, and invalidity contentions served on November 9, 2023). Petitioner’s prompt filing after serving invalidity contentions “weigh[s] against exercising the authority to deny institution.” *Fintiv*, Paper 11 at 11.

4. Overlapping issues with the parallel proceeding

There is no present overlap of prior art issues due to the early stage of district court litigation. Consequently, this factor favors institution.

5. Petitioner is a defendant

Petitioner is a defendant in the litigation. That is true of most Petitioners in IPR proceedings. Accordingly, this factor should not be a basis for denying institution.

6. Other circumstances

The prior art presented in this Petition provides compelling evidence of unpatentability of the Challenged Claims. This factor alone weighs against discretionary denial.

As such, because the *Fintiv* factors are either neutral or weigh against discretionary denial, and because this Petition was filed more than 3 months before the statutory bar date, institution should not be denied on discretionary factors.

X. CONCLUSION

Petitioner has established a reasonable likelihood that the Challenged Claims are unpatentable.

XI. MANDATORY NOTICES

A. Real Party-in-Interest

Pursuant to 37 C.F.R. §42.8(b)(1), Petitioner certifies that the real party-in-interest is SAP America, Inc. No other party had control over or access to the present Petition, and no other party was involved in developing the grounds presented in this Petition. In addition, no other parties funded the present Petition.

B. Related Matters

Pursuant to 37 C.F.R. §42.8(b)(2), to the best knowledge of the Petitioner, the '178 patent is or was involved in the following cases:

Case Heading	Number	Court	Filed
<i>Isix IP LLC v. International Business Machines Corporation</i>	<i>6:22-cv-00201</i>	W.D. Tex.	February 25, 2022
<i>Isix IP LLC v. Boomi, LP et al</i>	<i>6:22-cv-00275</i>	W.D. Tex.	March 14, 2022
<i>Isix IP LLC v. SAP America, Inc.</i>	<i>2:23-cv-00237</i>	E.D. Tex.	May 26, 2023

C. Lead and Back-up Counsel and Service Information

Lead Counsel	Backup Counsel
Jeffri A. Kaminski (Reg. No. 42,709) VENABLE LLP 600 Massachusetts Ave NW Washington, DC 20001	Frank C. Cimino, Jr. (Reg. No. 39,945) VENABLE LLP 600 Massachusetts Ave NW Washington, DC 20001

Tel: (202) 344-4048 Fax: (202) 344-8300 jakaminski@venable.com	Tel: (202) 344-4569 Fax: (202) 344-8300 fccimino@venable.com Megan S. Woodworth (Reg. No. 53,655) MSWoodworth@Venable.com VENABLE LLP 600 Massachusetts Ave., NW Washington, DC 20001 T 202-344-4507 F 202-344-8300
--	---

Please address all correspondence in this proceeding to lead and back-up counsel. Petitioner consents to service in this proceeding by email at the addresses above.

Dated: February 26, 2024

Respectfully submitted,

/Jeffri A. Kaminski/
Jeffri A. Kaminski
Lead Counsel for Petitioner
Registration No. 42,709
VENABLE LLP
600 Massachusetts Ave NW
Washington, DC 20001
Tel: (202) 344-4048
Fax: (202) 344-8300
Email: jakaminski@venable.com

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. §42.24(d), Petitioner hereby certifies, in accordance with and reliance on the word count provided by the word-processing system used to prepare this Petition, that the number of words in this paper is 13,993. Pursuant to 37 C.F.R. §42.24(d), this word count excludes the table of contents, table of authorities, mandatory notices under §42.8, certificate of service, certificate of word count, appendix of exhibits, and any claim listing.

Dated: February 26, 2024

/Jeffri A. Kaminski/
Jeffri A. Kaminski
Lead Counsel for Petitioner
Registration No. 42,709
VENABLE LLP
600 Massachusetts Ave NW
Washington, DC 20001
Tel: (202) 344-4048
Fax: (202) 344-8300
Email: jakaminski@venable.com

CERTIFICATE OF SERVICE

The undersigned certifies that, in accordance with 37 C.F.R. §42.6(e) and 37 C.F.R. §42.105, service was made on Patent Owner as detailed below.

Date of service February 26, 2024

Manner of service FEDERAL EXPRESS

Documents served The foregoing petition for *inter partes* review, together with all exhibits and other documents filed therewith.

Persons served Robert D. Null
BARNES & THORNBURG
600 1st Source Bank Center
100 North Michigan Street
South Bend, IN 46601

/Jeffri A. Kaminski/
Jeffri A. Kaminski
Lead Counsel for Petitioner
Registration No. 42,709
VENABLE LLP
600 Massachusetts Ave NW
Washington, DC 20001
Tel: (202) 344-4048
Fax: (202) 344-8300
Email: jakaminski@venable.com