Data Uncertainty Model for Mashup

Xin Gao
School of Electronics Engineering and Computer Science
Peking University
Beijing, China
gaoxin54@gmail.com

Wenhui Hu, Wei Ye, ZHANG Shi-kun
National Engineering Research Center for Software Engineering
Key laboratory of High Confidence Software Technologies (Ministry of Education)
Peking University
Beijing, China
xiaodiahu@gmail.com

Abstract—Mashup is a new kind integration application and users can compose related services as components to build new application—the mashups. Now the services on the web have different degrees of data uncertainty, including data error, stale data, and improper data processing and so on. We provide a data uncertainty model for mashup component which is assessed in the space of the homogeneous components that have the same functionality, and then we give the uncertainty computation mechanism of mashups based on the data uncertainty of mashup components and the computation sequence for different composition relationships.

Keywords-component; mashup; data uncertainty

I. INTRODUCTION

In Web development, a Mashup is a Web application that combines data from more than one source into a single integrated tool. The term Mashup implies easy, fast integration, frequently done by access to open API's and data sources to produce results data owners had no idea could be produced[1]. And mashups are applications developed by integrating content and functionality sourced from the Web. Mashups typically integrate heterogeneous elements available on the Web, such as RSS/Atom feeds, Web services, content scraped from third-party websites, or widgets (such as Google Maps)[2]. Different kinds of mashups reuse user interface (UI) components to build the composite application's UI, leverage and require external computational services, or simply integrate multiple plain data sources. So web environment has been required to provide data services with high quality.

With more and more sources of web service, the number and types of web service increase significantly. Now the services that we can get from the Internet include maps, e-mail, pictures, weather, traffic, search, etc. From the QWS Dataset [3,4,5] and ProgrammableWeb [6], the statistics of web service shows that there are a lot of choices in each type, and the overall amount of web service increases all the time. So we regard these services that share the same functionality as homogeneous services. There are thousands of web services, and also a large number of mashup services for the users, which build a solid base for related applications. From the point of view of the source of web service, it has changed from the original single site to numerous web application’s open APIs and service supporting architectures like SaaS, SOA, and Cloud computing. So the users can access to the original stand-alone local application in the form of service by the service framework like Axis, and also can access to the core data and applications through SaaS platforms and the OpenAPI of flickr, Google and yahoo on the web.

So user can use the mashup tools to build the new application, just so-called situational applications—that is, applications where the developer is also the final user and that serve a highly focused purpose. Situational applications typically aim to answer a precise query over a limited but heterogeneous data space. Their quality, therefore, depends strongly on the information that different integrated components can provide. As the promotion of mashup applications, there is one issue that has taken more attentions and is unavoidable: data uncertainty.

The uncertainty of the data source is very common on the web. We can see the web as the largest database of human society. This database is variable and uncertain because of numerous sources, complex structure, frequent change and dynamic transmission. The data uncertainty exists as inconsistent data, fault data, data variance, service instability and the impossible traceability of the change of data. We focus on the variance of web data which can be caused by error data source, unsuitable data process and stale data. So there are always many alternate results for the same data. For example, yahoo search service and Google search service have different results for the same keyword by a compare web site [7]. So the web can also be viewed as the largest uncertain database, and uncertain homogeneous service is an important part. For example, there are many services from travel sites which provide hotel information of scenic spots. But because of different information source and data update mechanism of these web sites, there may be some differences of price and room number among the services, and the uncertain data will mislead the travelers. Now most of service selection methods only focus on QoS strategy without considering the uncertainty aspect, so it can’t meet the requirement of providing users with reliable service in web environment. The mashup tools use these uncertain services as mashup components to build mashup applications, and it needs corresponding solutions for the data uncertainty of mashup. So data uncertainty is crucial for both components and composition in mashup. Assessing a mashup’s data uncertainty requires understanding both
component data uncertainty and the effect that the composition has on the final mashup’s overall data uncertainty.

In this paper, we introduce a data uncertainty model for mashup components, analyze different types of composition in mashup, and then define the calculation of data uncertainty for mashups based on the relationships of mashup components in the mashups. In the following parts of this paper, section 2 describes related work of mashup and data uncertainty. Section 3 describes the data uncertainty of mashup components, including the model to describe uncertain mashup components and the mechanism to access the data uncertainty of components. Section 4 introduces the composition of mashup components. Section 5 describes the data uncertainty of mashups, including the model describing the mashups and the mechanism to access the data uncertainty of mashups. Section 6 makes the conclusions and discusses the future work.

II. RELATED WORK

In traditional database applications, the existence and accuracy of the data is deterministic. In recent years, as technology advances, people deepen the understanding of data accuracy of the data is deterministic. In recent years, as the acquisition and processing technology. Now uncertain data has widely attracted public’s attention, and many researchers have focused on uncertainty in data processing. Literature [12] describes the background of uncertainty in the application of the data, and summarizes the challenges of uncertainty data management. Literature [13] gives an overview of algorithms and application of uncertain data management techniques. From the point of view of uncertain systems, Purdue University's Orion project [10] tries to design a general-purpose uncertain database system; the Trio project at Stanford University [11] studies the lineage according to the analysis of uncertainty. In the model aspect, the most commonly used model is possible world model [8, 9], which evolves a lot of database instances (called possible world instances) from uncertain database. There are also some researchers who focus on data integration, and literature [14] designs data-integration system which handles uncertainty at three levels: semantic mapping, uncertain data and uncertain queries. So based on these progresses of data processing on uncertain data, we can carry out the work to handle data uncertainty of web service regarded as the data source.

Currently there are a number of Mashup tools. Damia[15] is a Mashup tool provided by IBM, which allows the users to assemble data feeds from Internet and enterprise data sources. Yahoo Pipes [16] is a web-based tool provided by Yahoo. The users can build mashup applications by aggregating and manipulating data from web feeds, web pages, and other services. A pipe is composed of one or more modules, each one performing a single task like retrieving feeds from a web source, filter, sort or merge feeds. Popfly[17] is a web-based Mashup application by Microsoft. It allows the users to create a Mashup combining data and media sources. The Mashup is built by connecting blocks. Apatar[18] is a Mashup data integration tool that helps users to integrate desktop data with the web. Users install a visual job designer application to create integration schemas called DataMaps. MashMaker[19] is a web-based tool by Intel for editing, querying and manipulating web data. So we can see that many mashup tools focus on the integration of web data sources, and the mashups is the result of data integration of web data from the data aspect.

There are only a few researches on the data uncertainty of mashup application. The UQBE [20] is a mashup tool for non-programmers that supports query-by-example (QBE) over a schema made up by the user without knowing the schema of the original sources. Based on automated schema matching with uncertainty, the UQBE system returns the best confident results. MashRank [21] is a mashup tool that treats ranking as a first-class citizen in mashup construction, and allows for rankjoining Web sources with uncertain information. Both these two tools which consider the uncertainty problem just study the data uncertainty of the ranking data and schema matching, but don’t study the composition of uncertain data and the data uncertainty of mashups. So the research on data uncertainty in mashup is in the initial stage, and more and more people will take attention to this topic.

III. DATA UNCERTAINTY OF MASHUP COMPONENTS

On the web, there are many alternates for each kind service as the result shown in ProgrammableWeb, and we refer to them as homogeneous services. And these services can be used as the components of mashup, and data uncertainty of these homogeneous services is just the main data uncertainty of mashup components. We think that the data uncertainty of mashup component is relative to their alternate components, and the data uncertainty is considered in the space of all the mashup components which share the same functionality. We see these mashup components as the tuples in uncertain database, each of which has its information and its confidence possibility that reflects component’s uncertainty. We think that the data uncertainty of component is caused by component’s quality to provide data service and user’s distrust among all the alternate choices. So we can get the information of the confidence degree of the component based on user’s decision when the users participate in the selection of homogeneous components, which can give the uncertainty assessment of the component. Based on the assessments, we can build uncertain model for these homogeneous mashup components.

A. Uncertain component model

We consider the data uncertainty of mashup component from two aspects: the ability to provide data service and user’s confidence of component’s data. Before the data processing of the data uncertainty of mashup components, we first need to propose the syntax description of mashup component. Now services for mashup are mainly described by the basic information, such as service name, operation and corresponding input / output, etc. At the same time, we should also consider the quality of the component and the attributes which can describe QoS (quality of component). Then we can provide the components which meet user’s requirements based on QoS and user decision. Therefore we need a unified component description model with additional QoS information, and the service model is extended to following model:

Definition 1: According to the consideration of QoS, a component can be identified as a six-tuple set:

\[ MC=\{\text{ componentName}, \text{ Op}, \text{ Input}, \text{ Output}, \text{ Profile}, \text{ Qset}\} \]
Where ComponentName is the name of the component, Op is the operation to get corresponding data, Input / Output is the input and output of the operation, Profile describes component auxiliary information. And Qset is a set of attributes which represents the qualities of component to provide data service.

\[
\text{Qset}=\{\text{Response Time, Availability, Throughput, Reliability, Successability, Latency}\}
\]

QoS attributes help determine which of the available components is the best and meets users’ requirements. The meanings of the attributes are as follows:

- **a)** Response Time: Time taken to send a request and receive a response
- **b)** Availability: Number of successful invocations/total invocations
- **c)** Throughput: Total Number of invocations for a given period of time
- **d)** Reliability: Ratio of the number of error messages to total messages
- **e)** Successability: Number of response / number of request messages
- **f)** Latency: Time taken for the server to process request

Considering the problem of uncertain data management, we apply probabilistic database model to create uncertain service model. We wish to model probabilistic information using a probability space whose possible outcomes are all the conventional instances. The finiteness of D implies that there are only finitely instances \( I \in D \). By finite probability space we define a probability space \((D, P[\cdot])\) in which D is finite. We shall use the equivalent formulation of pairs \((U, p)\), where \( U \) is the finite subset of D and the probability assignment \( p: U \rightarrow [0,1] \) satisfies \( \sum_{w \in U} p(w) = 1 \) and \( \sum_{w \in U} p(w) = P[U] \). Therefore, we use probabilistic model to map data uncertainty by adding the probability attribute to the description of components. This uncertain component model is as follows:

Definition 2: An uncertain component model can be identified as a four-tuple set:

\[
\text{MC} = \{\text{ComponentName, N, Qset, P}\}
\]

Among them, ComponentName stands for the service; N is a number which stands for the times that this service has been chosen, and it is also the subjective information of user decision in the model; the Qset is the same as definition 2; the additional P represents the assessment of the credibility of each component. So \( D_S \) represents the domain of all the homogeneous components, and \( D_a \) represents the subset of \( D_S \), then satisfied:

\[
\sum_{MC \in D_a} p(MC) = P[D_a] \quad \text{and} \quad \sum_{MC \in D_S} P(MC) = 1
\]

**B. Uncertainty Component Model Building**

From the definition of uncertain component model, we can see that how to compute probability attribute is the core work, and it relates to the problem of what is certainty and what is uncertainty. In our view, the origin of certain in the web is credibility assessment by users. Mashup component is a kind of software application in web environment, and is a dynamic changing data carrier for numerous users’ ideas and thoughts. So for the component, the user is the root cause of its existence, and the user's knowledge and sense of service determines the credibility of the service. And from the further analysis, user awareness of the component’s credibility comes from user's intuitive understanding of component’s quality properties which represent the objective uncertainty of component and user’s requirements which represent the subjective uncertainty of component. Therefore, the service’s credibility can be evaluated based on the combination of these two kinds of information of component. In order to support uncertain data assessment of homogeneous components, we need to establish the appropriate process to meet the requirements of component registration and management, component accessing and uncertainty modeling. Our approach sets the probability attributes in uncertain service model as follows: we first calculate the weight of each quality attribute according to statistics on the user's selection through the neural network approach; and then calculate quality of component; finally, according to the quality of components we get the credibility of uncertain component model which is the value of corresponding probability property.

Set up a group of homogeneous components \( \text{MC} = \{mc_1, mc_2, mc_3 ... mc_n\} \), and users give the choice based on personal feeling. Then we can use the number of choices of homogeneous components from the user group to calculate the services QoS attribute weights by neural network.

First, we need to collect and organize the user’s choice of homogeneous components. Within a certain period of time the user’s choice is a collection of components, which can be expressed as \( \text{UserChosen} = \{N_{i_1}, N_{i_2}, N_{i_3} ... N_{i_m}\} \), where \( N_{i_k} \) means that the \( i_k \) component has been chosen \( N_{i_k} \) times. Use matrix to organize QoS attributes of selected components as follow. It is the matrix in which each row represents a single component metric, where each column represents a single QoS attribute and the QoS attributes include Response Time, Availability, Throughput, Successability, Reliability and Latency.

\[
\text{QosM}=
\begin{bmatrix}
RT_1 & A_1 & T_1 & S_1 & R_1 & L_1 \\
RT_2 & A_2 & T_2 & S_2 & R_2 & L_2 \\
... & ... & ... & ... & ... & ... \\
RT_k & A_k & T_k & S_k & R_k & L_k \\
\end{bmatrix}
\]

Then set \( \text{QosM}^T \), which is a \( 6*n \) matrix, as the input of neural network algorithm unit which means that set every QoS attribute as a node in the neural network and the corresponding value of each component is an assignment of the node; set the times of each component being selected in the form of \( [N_{1_1}, N_{1_2}, ..., N_{m_1}] \), which is a \( 1*n \) matrix, to be the target; at last, construct the training sample. Through the training of neural network, we can get the weight for each QoS attribute. Then each QoS weight divides the max of them, and we can get the set of relative weights:

\[
W = \{W_1, W_2, W_3, W_4, W_5, W_6\}
\]

which are corresponding to the six attributes of QoS. Regarding the evaluation mechanism of service credibility, we view homogeneous components as an independent set and
compute relative quality of each component in the corresponding homogeneous component set as follow:

\[ \overline{RT}_i = 1 - \frac{RT_i}{\sum_{k=1}^{m} RT_k} ; \quad \overline{A}_i = \frac{A_i}{\sum_{k=1}^{m} A_k} \]

\[ \overline{T}_i = \frac{T_i}{\sum_{k=1}^{m} T_k} ; \quad \overline{R}_i = \frac{R_i}{\sum_{k=1}^{m} R_k} \]

\[ \overline{S}_i = \frac{S_i}{\sum_{k=1}^{m} S_k} ; \quad \overline{L}_i = \frac{L_i}{\sum_{k=1}^{m} L_k} \]

Because response time and latency are smaller, the quality of component’s data is better, so they are represented by their complements. Thus quality of component is formed by these relative qualities and the relative weights, just as follow:

\[ Q_i = \overline{Q}_i \cdot W^T = \overline{RT}_i \cdot W_1 + \overline{A}_i \cdot W_2 + \overline{T}_i \cdot W_3 + \overline{R}_i \cdot W_4 + \overline{S}_i \cdot W_5 + \overline{L}_i \cdot W_6 \]

When we get the relative quality \( Q_i \), the degree of credibility which is also the attribute \( P_i \) of component can be calculated by

\[ P_i = \frac{Q_i}{\sum_{k=1}^{m} Q_k} , \quad 1 \leq i \leq m \]

IV. COMPOSITION IN MASHUP

Assessing each mashup component’s data uncertainty isn’t enough: the final mashup application’s data uncertainty also depends on how these components are interconnected. We can assess the final applications’ overall data uncertainty by aggregating the composing services’ data uncertainty. Mashup’s data uncertainty isn’t simply an aggregation of individual component’s data uncertainty. Instead, it depends on how particular components combine into a composite logic, layout, and hence user experience. By analyzing the most popular mashups published on programmableweb.com, the paper [2] identified the following typical roles:

1) Master. Even if a mashup integrates multiple components in a single page, in most cases, one component is more important than the others.

2) Slave. A slave component’s behavior depends on another component: its state is mainly modified by events originating in another (master) component.

3) Filter. Filter components let users specify conditions over the content the other components show. They provide (possibly hierarchical) access mechanisms that let users incrementally select which content they want to see.

Based on these three roles, there are three basic patterns that characterize most mashup applications (see Figure 3) and highlight some mutual dependencies among the identified roles that impact mashup’s data uncertainty.

1) The slave-slave pattern, in which the mashup integrates several slave components the user can interact with in an isolated fashion, without any propagation of data or events from one component to another. At startup or during runtime, users define filter conditions that steer all the slave components. The effect is that of a rather static application with very simple interaction facilities that lets users “query” the slave components’ data set. Regarding the resulting mashup’s data uncertainty, we assume that the filter doesn’t increase the components’ data uncertainty.

2) The master-slave approach, the most widely used pattern among today’s mashup applications. It features all three component roles. A filter component lets users restrict the data all the other components simultaneously show. Users employ the master component to perform the main interactions with the application, such as selecting related data items. The slave component is automatically synchronized according to the selections performed on the master component, thereby visualizing the selected elements’ details. With the master-slave pattern, the final application’s data uncertainty could depend on the application’s composition logic. Provided that master and slave are compatible in terms of data to be visualized, their integration might increase the slave’s data uncertainty.

3) The master-master pattern. This is the most complete pattern, in which — in addition to suitable filter components — all integrated components are masters. All components provide interaction facilities that let users perform selections or that provide inputs that propagate to all the other components that synchronize accordingly. The master components therefore also act as slaves. From a data uncertainty perspective, the master-master pattern is similar to the master-slave pattern. If the components have different underlying data sets, situations could occur in which one component satisfies the user request, while another component can’t, lowering the mashup’s overall perceived data uncertainty.

We will give our data uncertainty computation mechanism based on these three roles of mashup components and the three mashup patterns. In this paper, we assume that the mashup composition performs integration at the process and presentation levels correctly. To characterize data uncertainty in the context of mashups, we focus on the data level.

V. DATA UNCERTAINTY OF MASHUPS

A. Date Uncertainty in the data sets of mashups

Data integration in mashups corresponds to a global-as-view (GAV) problem, in which the global schema is expressed in terms of views over the integrated data sources. During mashup development, the designer can inspect the attributes the components expose, as specified in the component APIs, and infer join attributes on which to base data integration.

Figure 1. Data sets involved in data integration of mashup.
We can characterize data integration for mashups by categorizing the data relationship between different data sources as follows:

- Mashup applications are developed to let users retrieve and access a set of data that we call the ideal data set (IDS); it is the final data view of mashup components.
- Each component \( k \) has its own data set \( D_{Sk} \). To fulfill the mashup requirements, a smaller portion \( SDS_k \subseteq D_{Sk} \) could be sufficient. \( SDS_k \) is the corresponding components’ situational data set.
- The integration of all situational data sets \( SDS_k \) gives the real data set \( RDS = \cup_i SDS_i \) that the mashup provides. \( RDS \)'s data uncertainty thus depends on the uncertainty of the data individual components provide.
- We can determine the mashup’s data uncertainty by comparing its RDS with the corresponding IDS.

Now we analyze the corresponding data uncertainty for RDS based on the forward work on data uncertainty of mashup components and their composition relationships.

- The data uncertainty of situational slave data set is the same as the data uncertainty of slave component. And we set it as \( P(S) \).
- The data uncertainty of situational master data set is the same as the data uncertainty of master component. And we set it as \( P(M) \).
- For the missing data is not the core data part of RDS and always have no strong relations with the slave component or the master component, now we don’t consider its data uncertainty.

Then the data uncertainty of RDS is calculated based on the type of Join relationship among data sets and the structure of mashup components. We assume components are sourced from the Web, we also assume they’re independent of each other. Because our data uncertainty is computed for the data source among the data space which is composed of the data sources which belong to the same domain, like weather, news, sales and so on, the data uncertainty of slave component and that of master component are independent. So the data uncertainty of join relationship between slave component and master component is as follow:

\[
P(M_S) = P(M) \times P(S)
\]

\[
P(U|CM) = 1 - P(S)
\]

We describe data uncertainty of data source with the probability of the data source to be certain. So \( P(M) \) is not only the probability of master component to be certain, but also the data uncertainty of master component; \( P(S) \) is the data uncertainty of slave component; \( P(MS) \) is the data uncertainty of the join data set between master component and slave component. \( P(U|CM) \) is the probability that the slave component is uncertain when the master component is certain. And we will describe different ways to compute \( RDS \)'s data uncertainty for different composition situations in mashups.

B. Data uncertainty of RDS

In the Slave-slave pattern, every mashup component is independent with each other, and the data uncertainty of mashup components is the maximum data uncertainty of all the slave components. Because we describe data uncertainty of data source with the probability of the data source to be certain, the data uncertainty of mashup components is described by \( P\{S_1, S_2, ..., S_n\} \).

\[
P\{S_1, S_2, ..., S_n\} = \min\{P(S_1), P(S_2), ..., P(S_n)\}
\]

For the Master-slave pattern and Master-Master pattern, the data uncertainty is mainly to compute the data uncertainty between master component and slave component, because of the data uncertainty between master component and master component can be transformed to data uncertainty between master component and slave component. In the situation of Master-slave pattern, there are two sources of data uncertainty which are the one that the master component is uncertain and the other that the slave component is uncertain when the master component is certain. So the data uncertainty between master component and master component and master component is as follow:

\[
P(M_S) = 1 - (P(U|M) + P(U|CM)) = P(M) - P(U|CM) = P(M) + P(S) - 1
\]

Where \( P(M_S) \) means data uncertainty between a master component and a slave master component, \( P(U|M) \) means the probability that master component is uncertain and \( P(U|CM) \) means the probability that the slave component is uncertain when the master component is certain.

We can consider Master-Master pattern to be the combination of two Master-slave patterns: a selection in one master causes the other master to act as a slave and vice versa.

\[
P(M_M) = 1 - (\alpha P(U|M_1) + P(U|CM_1)) + (1 - \alpha)(P(U|M_2) + P(U|CM_2)) = P(M_1) + P(M_2) - 1
\]

Where \( P(M_M) \) means data uncertainty between two master components; \( P(U|M) \) means the probability that master component is uncertain; \( P(U|CM) \) means the probability that one master component is uncertain when another master component is certain.

In real mashups, we can analyze the mashups into the three patterns, but we also need to set the data uncertainty computation sequence for the mashups. For the computation sequence, we have set three rules:

- Master-Master pattern is computed prior to others.
- Master-Slave pattern is computed prior to Slave-Slave pattern.
- The pairs of mashup components which have smaller data uncertainty are computed prior.

![Figure 2. The situations of computation sequence](image)

Based on master role and slave role of mashup components, we categorize the computation sequence into five situations:
Situation 1: Slave-Slave-Master-Master
Because there are Master-Master pattern, we compute the pair of Master-Master first. Then this pair play the master role to the second slave component, we compute them as Master-Slave pattern. At last, we compute the forward result and the first slave component as Slave-Slave pattern. The computation sequence is as follow:

\[ P(\text{slave-slave-Master-Master}) = P( \text{slave-(slave-(Master-Master)))} \]

Situation 2: Master1-Master2-Mater3
In this situation, we first find the master component which has the lowest data uncertainty. If Master1 or Master3 has the lowest data uncertainty, computation sequence is from the left to the right or from the right to the left. If the middle one has the lowest data uncertainty, then find the one between the other two components which has lower data uncertainty and compute this pair first.

Situation 3: Slave1-Master-Master-Slave2
This situation can also be seen as Slave1-Master-Slave2, and the computation sequence depends on the data uncertainty of Slave1 and Slave2. If the data uncertainty of Slave1 is higher than the data uncertainty of Slave2, the computation sequence is as follow:

\[ P(\text{Slave1-Master-Master-Slave2}) = P( \text{Slave1- ((Master-Master)-Slave2}) \]

where \( P(\text{Slave1}) > P(\text{Slave2}) \). The situation first executes Master-Master computation and then executes Master-Slave computation.

Situation 4: Master-slave-slave-Master
In this situation, the computation sequence first executes Master-Slave computation and then executes Slave-Slave computation.

\[ P(\text{Master-slave-slave-Master}) = P( \text{(Master-slave)-(slave-Master)}) \]

Situation 5: Master1-slave-Master2
In this situation, there are two Master components and we assume that the data uncertainty of Master1 is higher than the data uncertainty of Master2. If Master1 and Master2 communicate separately with different parts of Slave component, we can transform the situation to Master1-slave1-slave2-Master2 situation where slave1 and slave2 are different parts of Slave component and \( P(\text{slave1}) > P(\text{slave2}) = P(\text{Slave}) \). If the parts of Slave component that Master1 and Master2 communicate separately with have the common part, the computation sequence is as follow:

\[ P(\text{Master1-slave-Master2}) = P(\text{Master1-(slave-Master2)}) \]

VI. CONCLUSION
In this paper, we study the data uncertainty computation of mashups based on these relationships and we show the computation sequence in different composition situations. We will further our work for the complex integration process in mashup application, analyze how the change of data uncertainty in certain mashup component affects the overall data uncertainty of mashups, and study the composition of mashup applications based on data uncertainty.

ACKNOWLEDGMENT
This work was supported by the National Natural Science Foundation of China under Granted No. 60903001

REFERENCES